Table of contents

```
                                        .TITLE  TSEM4   TSX-Plus EMT Overlay
                                        .ENABL  LC
                                        .ENABL  AMA
                                        .DSABL  GBL
                                ;
                                ;  Copyright (C) 1976,1977,1978,1979,1980,1981,1982,1983,1984,1985,1986,
                                ;                1987,1988,1989.
                                ;
                                ;  S&H Computer Systems, Inc.
                                ;  Nashville, Tennessee
                                ;
                                ;  This software is furnished under a license for use only
                                ;  on a single computer system and may be copied only with
                                ;  the inclusion of the above copyright notice.  This
                                ;  software, or any other copies thereof, may not be provided
                                ;  or otherwise made available to any other person except
                                ;  for use on such system and to one who agrees to these
                                ;  license terms.  Title to and ownership of the software
                                ;  shall at all times remain with S&H Computer Systems, Inc.
                                ;
000000                                  .CSECT  TSEM4
000000  020552                  TSEM4:  .RAD50  /EM4/            ;Overlay id
        177776                  PS      =       177776          ;Processor Status Word
                                ;
        000000                  EC0     = 0                     ;EMT error code return values
        000001                  EC1     = 1
        000002                  EC2     = 2
        000003                  EC3     = 3
        000004                  EC4     = 4
        000005                  EC5     = 5
        000006                  EC6     = 6
        000007                  EC7     = 7
        000010                  EC8     = 8.
        000011                  EC9     = 9.
        000012                  EC10.   = 10.
                                ;
                                ;  Macro calls
                                ;
                                        .MCALL  .READW,.CLOSE,.CRRG
                                ;
                                ;  Global definitions
                                ;
                                        .GLOBL  KMNEMT,DETEMT,PMEMT,SPLEMT,SSEMT,KILJOB
                                        .GLOBL  CLEMT,CLCLOS,VLSEMT
                                ;
                                ;  Global references
                                ;
                                        .GLOBL  WC$LEN,WC$NPR,RC$LEN
                                        .GLOBL  WC$$SZ,WCBBAS,WC$SIZ,WC$OFF,WC$PAR,WC$TRP,SR$WCB
                                        .GLOBL  NUMWCB,RCBBAS,RCBEND,SHRRCB,SHRRCN,RC$FLG,RC$DSP,RC$LCG
                                        .GLOBL  MAXSRD,SR$PAR,SR$PDR,SR$PX,SR$FLG,SR.MOR,SR.DSP,DOTRMP
                                        .GLOBL  $VBELL,IB$SF2,SF2LEN,GETQ,IB$IJ,SPPRED,$VNOTT,LCXTBL
                                        .GLOBL  $NOVLN,$SUCF,$LOFCF,$NOIN,MAXSEC,LSECPT,FSTSL,LSTSL,INITLN
                                        .GLOBL  LSW6,CSHALC,CSHINI,SPLDEL,SETDTR,CO$DTR,CLWTIM,$RBRK,LINSWT
                                        .GLOBL  $IOMAP,R.GSTS,INTPRI,CURRDB,UPARO,RF$WRT,RT$FLG,LPROG
                                        .GLOBL  $INKMN,LSW4,BADEMT,EMTBLK,LOGOFF,SETHAN,REBOOT
                                        .GLOBL  KMCPCX,EMSPHL,EMSPNH,GTUKBL,LEMTPC,CLREST,$RDSAV,LSW11
```

```
                        .GLOBL  EMTLEV,LJSW,JSWLOC,PRGSIZ,SUTOP,CXTRMN,R$CHN,CLCLER
                        .GLOBL  CHNSIZ,R$XCHN,RUNCHN,PRGTOP,CSIARE,OVLBIT,NEWJSW
                        .GLOBL  CHAIN,CINFLG,UERSEV,ERRLOC,RUNARG,CINDAT,CPLEMT,RMNBAS
                        .GLOBL  VIMAGE,$VIRJB,LSW9,RMON,LSW2,$SCOPE,TO$SCP,$TAB
                        .GLOBL  TO$TAB,$FORM,TO$FF,TO$LC,CXTRMN,R$TTOP,TTOPTS,RMNBAS
                        .GLOBL  $SETRN,USRSTK,$RNIOP,$IOMAP,SETMAP,$RNMLK,LQUAN,JSTK
                        .GLOBL  UMODE,UPMODE,USTART,$DEBUG,DBGENT,ABORT,SUPRTN,VIDCSR
                        .GLOBL  VO$RDB,R.GSIZ,RUNRDB,VO$WDB,VO$WDE,R.GID,RC$BAS
                        .GLOBL  W.NRID,W.NOFF,KPAR0,W.SIZE,SETERR,NSPLDV
                        .GLOBL  OVRHC,SPOLGO,EMTXIT,$DILUP,LSW,$SUSPN,FORCEX,$DETCH
                        .GLOBL  LNPRIM,CL$XLN,CM$BRK,CM$ON,CL$STA,LXCL,CLXICP,LINIR
                        .GLOBL  S$SPND,QNSPND,LSTPL,CLXBRK,$CTRLS,LSW3,$CTRLO
                        .GLOBL  ENQTL,S$CPU,LOTSPC,LOTSIZ,PO$BYP
                        .GLOBL  $DISCN,TRNSTR,PO$DET,PRIVCO,FSTDL,LSTDL,VALADB,LSUCF
                        .GLOBL  DETCBS,GETUCH,CORUSR,LPARNT,URO,RPAR,RPDR,VPLAS
                        .GLOBL  MAPPLS,VALADW,RDB,RDBEND,RT$NAM,RT$$SZ,RT$BAS,RT$TOP
                        .GLOBL  CUPAR0,CUPDR0,UPDR0,CS$OPN,CHNADR,C.CSW,NSPLDV,SPLSHF
                        .GLOBL  P2$TRM,PRIVC2,CLTOTL,LSTHL,$HARD,LCLUNT,$DEAD,SPLSFF
                        .GLOBL  FSTIOL,LSTIOL,CL$LIX,CL$RQH,CL$WQH,LCLUNT,NEDCHR,SPLSWF
                        .GLOBL  LOUTIR,TTINCP,$AUTO,ILSW2,S9600,SETSPD,LINRTS,CLOTIR,CLINCP
                        .GLOBL  $SXON,LSW10,CO$DEF,$TAB,CO$TAB,CO$FF,$8BIT,CO$8BT
                        .GLOBL  CL$OPT,CL$STA,CL$COL,CL$LEN,CL$LIN,CL$SKP,CL$WID
                        .GLOBL  CL$EPN,CL$EPS,CL$EPP,CS$SPL,CS$NMX,CLDEVX,C.DEVQ,C1DEVX
                        .GLOBL  CL$ORS,CL$ORA,CM$EFP,$XCHAR,PMCELS,PMUSER,PMRUN,PMBASE,PMTOP
                        .GLOBL  PMFLGS,PMCELS,PMNBPC,ENSYS,FP$MOV,VPAR6,PMPAR,KPAR6
                        .GLOBL  P2$WRL,LPROJ,P2$GRP,P2$SAM,VSWPFL,DFJMEM
                        .GLOBL  $UDSPC,ODTBAS,SR3MMR,USDSPC,IDSFLG
                        .GLOBL  SETSTN
                ;
                ;-------------------------------------------------------
                ;  Macros to enable and disable interrupts.
                ;
                        .MACRO  DISABL          ;DISABLE INTERRUPTS
                        BIS     #340,@#PS
                        .ENDM   DISABL
                ;
                        .MACRO  ENABL           ;ENABLE INTERRUPTS
                        BIC     INTPRI,@#PS
                        .ENDM   ENABL
                ;
                ;  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 calling 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
```

```
                                        .SBTTL  TSKMON EMT's
                                ;-------------------------------------------------------------------------
                                ;  KMNEMT is jumped to when a KMON specific EMT is executed.
                                ;  It interprets the sub-function (in 2nd word of arg block)
                                ;  and branches off to the appropriate processing routine.
                                ;
000002  032761  000000G 000000G KMNEMT: BIT     #$INKMN,LSW4(R1);IS KMON RUNNING NOW?
000010  001002                        BNE     1$              ;BR IF YES
000012  000137  000000G         2$:     JMP     BADEMT          ;ILLEGAL EMT IF KMON NOT RUNNING
                                ;
                                ;  Get sub-function code and branch off to processing routine.
                                ;
000016  013702  000020G         1$:     MOV     EMTBLK+2,R2     ;GET SUB-FUNCTION CODE
000022  020227  000022                  CMP     R2,#KMNMAX      ;VALID SUB-FUNCTION CODE?
000026  103371                        BHIS    2$              ;ERROR IF NOT
000030  006302                        ASL     R2              ;CONVERT TO WORD TABLE INDEX
000032  000172  000036'              JMP     @KMNSUB(R2)     ;ENTER PROCESSING ROUTINE

                                ;-------------------------------------------------------------------------
                                ;  Table of KMON subfunctions for EMT code 126.
                                ;  (Sub-function code is stored in 2nd word of EMT arg block)
                                ;
000036  000102'                 KMNSUB: .WORD   KMRUN            ; 00 Initiate execution of a SAV file
000040  001362'                         .WORD   KMSPLG          ; 01 Initiate a spooler
000042  000000G                         .WORD   LOGOFF          ; 02 Log off the current job
000044  000000G                         .WORD   SETHAN          ; 03 Update running copy of handler
000046  000000G                         .WORD   REBOOT          ; 04 Reboot the system
000050  001752'                         .WORD   KILEMT          ; 05 Force logoff of a job
000052  002122'                         .WORD   CSHNEW          ; 06 Clean out data cache
000054  002140'                         .WORD   KMSPDL          ; 07 Delete a spool file
000056  001406'                         .WORD   KMSPEK          ; 10 Get data from within system
000060  001436'                         .WORD   KMSPOK          ; 11 Store data into system
000062  001466'                         .WORD   KMSPRS          ; 12 Suspend or resume a job
000064  001540'                         .WORD   CLXCON          ; 13 Cross connect TT line to CL unit
000066  000000G                         .WORD   KMCPCX          ; 14 Copy data from context blk
000070  000000G                         .WORD   EMSPHL          ; 15 Set HOLD mode for a spooler
000072  000000G                         .WORD   EMSPNH          ; 16 Set NOHOLD mode for a spooler
000074  000000G                         .WORD   GTUKBL          ; 17 Access block in user key region
000076  001674'                         .WORD   STTTLN          ; 20 Start up a terminal line
000100  000000G                         .WORD   SETSTN          ; 21 Set site name for flag pages (in TSSPL2)
000022          KMNMAX  =       <.-KMNSUB>/2
```

```
                                        .SBTTL  KMRUN  -- Start execution of a SAV file
                                ;-------------------------------------------------------------------------
                                ;  KMRUN (126,0) is the EMT executed by KMON to start execution of
                                ;  a SAV file.  On entry user channel # RUNCHN must be open to the SAV file.
                                ;  The SAV file is read into memory and control is transfered to it.
                                ;
000102  005061  000000G        KMRUN:  CLR     LEMTPC(R1)          ;CLEAR ADDRESS OF LAST USER-MODE EMT
000106  042761  000000G 000000G        BIC     #$INKMN,LSW4(R1);SAY WE ARE EXITING FROM KMON
000114  012737  177777  000000G        MOV     #-1,EMTLEV          ;SAY NO EMT IS BEING DONE
                                ;
                                ;  Set up JSW for job so virtual-image flag will be set before mem alloc
                                ;
000122  013746  000000G                MOV     NEWJSW,-(SP)        ;SET NEW JSW
000126  011661  000000G                MOV     (SP),LJSW(R1)
000132  106637  000000G                MTPD    @#JSWLOC
                                ;
                                ;  Set $RDSAV flag to prevent PKSTAT/UPSTAT from saving/restoring the
                                ;  JSW and Error byte in locations 44 and 52 of memory.  This is necessary
                                ;  because SUTOP is going to change mapping to an uninitialized area
                                ;  of memory and we don't want to use the contents of memory locations
                                ;  until we actually get the SAV image in and set it up.
                                ;
000136  052761  000000G 000000G        BIS     #$RDSAV,LSW11(R1);Say we are reading in SAV file
                                ;
                                ;  Set top of memory for the sav file we are about to start.
                                ;
000144  013700  000000G                MOV     PRGSIZ,R0           ;SET TOP OF MEMORY FOR JOB REGION
000150  105737  000000G                TSTB    VSWPFL              ;IS THIS A SWAPPING SYSTEM?
                                ;;      BNE     39$                 ;IF SWAPPING USE JOB SIZE, ELSE
                                ;; Changing this to always use PRGSIZ allows RT-11 utilities to
                                ;; work correctly in a 64kb DFLMEM no swapping system.  They use
                                ;; .GVAL offset 266 to determine base of USR (we set this in SUTOP
                                ;; to 177774 if DFLMEM is 64kb) then TST -(R0) (which traps because
                                ;; we only map PAR7 through the top of simulated RMON), prior to
                                ;; doing a .SETTOP.  If the program is virtual (doesn't map RMON),
                                ;; then PRGSIZ will be bigger than 56kb; if not, then pretend
                                ;; that USR base is below RMON.  See TSKM2C CMDRUN for PRGSIZ set up.
000154  000406                        BR      39$                 ;; Always use PRGSIZ as set up by TSKMON
000156  013700  000000G                MOV     DFJMEM,R0           ;RETRIEVE DEFAULT MEMORY SIZE
000162  000300                        SWAB    R0                  ;CONVERT TO ADDRESS ( * 256. )
000164  006300                        ASL     R0                  ;   * 512.
000166  005300                        DEC     R0                  ;LEAVE 1 WORD SHORT (ELSE 64.KB = 0)
000170  006300                        ASL     R0                  ;   * 1024.  = ALLOCATED TOP ADDRESS IN BYTES
000172  004737  000000G        39$:    CALL    SUTOP               ;SET TOP OF MEMORY
                                ;
                                ;  Transfer the channel information from internal channel RUNCHN
                                ;  into user's channel # 17
                                ;
000176  013700  000000G                MOV     CXTRMN,R0           ;Get address of job's simulated RMON
000202  010003                        MOV     R0,R3
000204  062703  000000C                ADD     #R$CHN+<17*CHNSIZ>,R3 ;Point to chan blk for channel 17
000210  010002                        MOV     R0,R2
000212  062702  000000C                ADD     #R$XCHN+<<RUNCHN-21>*CHNSIZ>,R2 ;Point to RUNCHN block
000216  010200                        MOV     R2,R0               ;Save pointer to RUNCHN block
000220  012704  000000C                MOV     #CHNSIZ/2,R4        ;Get # words to copy
000224  012223                4$:     MOV     (R2)+,(R3)+         ;Copy RUNCHN to channel 17
000226  077402                        SOB     R4,4$
```

```
                                        CLR     (R0)            ;Say RUNCHN is closed
                                ;
                                ;  Read the SAV file into memory
                                ;
                                        MOV     PRGTOP,R3       ;GET HIGHEST ADDRESS IN SAV FILE IMAGE
                                        CLC
                                        ROR     R3              ;GET # WORDS TO READ
                                        INC     R3              ;CVT FROM TOP ADDRESS TO # WORDS
                                        .READW  #CSIARE,#17,#0,R3,#0
                                        BCC     16$             ;Br if ok
                                        JMP     9$              ;Error during read
                                ;
                                ;  If program is not overlayed, close channel 17.
                                ;
                                16$:    BIT     #OVLBIT,NEWJSW  ;IS THE PROGRAM OVERLAYED?
                                        BNE     3$              ;IF YES THEN LEAVE CHANNEL 17 OPEN
                                        .CLOSE  #17             ;CLOSE CHANNEL 17
                                ;
                                ;  See if we need to allocate extended-memory region for virtual overlays
                                ;
                                3$:     CALL    KMRVM           ;See if we need to allocate XM region
                                ;
                                ;  If a .CHAIN was done, move data from context block to chain data area.
                                ;
                                        MOV     NEWJSW,R3       ;GET JSW FOR JOB BEING STARTED
                                        BIC     #CHAIN,NEWJSW   ;CLEAR CHAIN FLAG
                                        TSTB    CINFLG          ;WAS A .CHAIN DONE?
                                        BEQ     20$             ;BR IF NOT
                                        CLRB    CINFLG          ;SAY CHAIN IS FINISHED
                                        BIS     #CHAIN,NEWJSW   ;SET CHAIN-FLAG
                                        BIT     #CHAIN,R3       ;DOES JOB WANT TO PRESERVE CHAIN DATA AREA?
                                        BNE     1$              ;BR IF YES
                                        MOV     #<<1000-500>/2>,R4 ;GET # WORDS TO MOVE
                                        CALL    CINMOV          ;Move chain data
                                ;
                                ;  If a .CHAIN was done, reset the previous user's error cell (52&53).
                                ;
                                        MOVB    UERSEV,-(SP)    ;GET THE PREVIOUS ERROR SEVERITY LEVEL
                                        SWAB    (SP)            ;PUT INTO HIGH BYTE
                                        MTPD    @#ERRLOC        ;RESTORE THE USER ERROR SEVERITY LEVEL
                                        BR      6$              ;SKIP THE INSTRUCTIONS TO CLEAR ERRLOC
                                ;
                                ;  We are not doing a .CHAIN.
                                ;  See if we need to pass run argument string through chain data area.
                                ;
                                20$:    TSTB    RUNARG          ;See if we need to pass RUN command line
                                        BNE     22$             ;Br if yes
                                        MOV     #1,CINDAT+10    ;Say argument length = 1
                                        CLR     CINDAT+12       ;Say argument string is null
                                22$:    BIT     #CHAIN,R3       ;Does job want to preserve chain data area?
                                        BNE     1$              ;Br if yes
                                        MOV     CINDAT+10,R4    ;Get # bytes in run argument string
                                        ADD     #13,R4          ;Get # bytes below 512 and round up
                                        ASR     R4              ;Get # words to move
                                        CALL    CINMOV          ;Move run argument to chain data area
                                ;
                                ;  Initially clear user's error cell (52&53).
```

```
                                        ;
000456  005046                  1$:     CLR     -(SP)
000460  106637  000000G                 MTPD    @#ERRLOC        ;INITIALLY ZERO USER'S ERROR CELL
                                        ;
                                        ;  Store [EMT 356] instruction in job's low memory area to be used
                                        ;  to exit from a completion routine.
                                        ;
000464  012746  104356            6$:     MOV     #104356,-(SP)   ;[EMT 356]
000470  006637  000000G                 MTPI    @#CPLEMT        ;STORE INTO JOB'S AREA
                                        ;
                                        ;  Set pointer to simulated monitor vector table that is mapped through
                                        ;  user page 7 (160000-177777).
                                        ;
000474  012746  000000G                 MOV     #RMNBAS,-(SP)   ;VIRTUAL ADDRESS OF MONVEC TABLE
000500  032737  000000G 000000G         BIT     #VIMAGE,NEWJSW  ;IS THIS A VIRTUAL JOB?
000506  001405                        BEQ     14$             ;BR IF NOT
000510  052761  000000G 000000G       BIS     #$VIRJB,LSW9(R1);SET VIRTUAL-JOB FLAG FOR THE SYSTEM
000516  012716  177776                MOV     #177776,(SP)    ;SET THIS AS RMON BASE FOR VIRTUAL JOB
000522  106637  000000G          14$:   MTPD    @#RMON          ;SET POINTER IN USER'S SPACE
                                        ;
                                        ;  Set EMT vector cell (location 30) in user's area to point to
                                        ;  cell in simulated RMON vector.
                                        ;
000526  005000                        CLR     R0              ;INITIALIZE OPTION WORD VALUE
000530  016102  000000G               MOV     LSW2(R1),R2     ;GET VALUE OF LSW2 FLAGS FOR LINE
000534  032702  000000G               BIT     #$SCOPE,R2      ;IS THIS A SCOPE TYPE TERMINAL?
000540  001402                        BEQ     10$             ;BR IF NOT
000542  052700  000000G               BIS     #TO$SCP,R0      ;SET SCOPE FLAG
000546  032702  000000G          10$:   BIT     #$TAB,R2        ;HARDWARE TABS?
000552  001402                        BEQ     11$             ;BR IF NOT
000554  052700  000000G               BIS     #TO$TAB,R0      ;SET TAB FLAG
000560  032702  000000G          11$:   BIT     #$FORM,R2       ;HARDWARE FF?
000564  001402                        BEQ     12$             ;BR IF NOT
000566  052700  000000G               BIS     #TO$FF,R0       ;SET FORM-FEED FLAG
000572  032702  000000G          12$:   BIT     #TO$LC,R2       ;ALLOW LOWER-CASE INPUT?
000576  001402                        BEQ     13$             ;BR IF NOT
000600  052700  000000G               BIS     #TO$LC,R0       ;SET LC FLAG
000604  013702  000000G          13$:   MOV     CXTRMN,R2       ;GET ADDRESS OF BASE OF SIMULATED RMON FOR JOB
000610  010062  000000G               MOV     R0,R$TTOP(R2)   ;SET TT OPTION WORD VALUE
000614  012746  000000C               MOV     #TTOPTS+RMNBAS+2,-(SP);GET ADDRESS OF TT OPTION CELL
000620  106637  000030                MTPD    @#30            ;SET IN EMT TRAP VECTOR
                                ;
                                ;  See if we need to some special initialization for SETUP program
                                ;
000624  032761  000000G 000000G         BIT     #$SETRN,LSW9(R1);Is this the SETUP program being started?
000632  001402                        BEQ     19$             ;Br if not
000634  004737  001024'               CALL    SETINI          ;Initialize for SETUP program
                                ;
                                ;  Set up user's SP
                                ;
000640  013746  000000G          19$:   MOV     USRSTK,-(SP)    ;STACK POINTER FOR PROGRAM BEING STARTED
000644  106606                        MTPD    SP              ;SET AS USER-MODE SP
                                ;
                                ;  Set up JSW for job in memory now that SAV image has been read in
                                ;
000646  013746  000000G               MOV     NEWJSW,-(SP)    ;SET NEW JSW
000652  011661  000000G               MOV     (SP),LJSW(R1)
```

```
000656  106637  000000G                 MTPD    @#JSWLOC
                                ;
                                ;  Now that memory image is set up correctly, clear the flag that says
                                ;  we are reading in the SAV file.
                                ;
000662  042761  000000G 000000G         BIC     #$RDSAV,LSW11(R1);Finished reading in SAV file
                                ;
                                ;  See if we should map PAR 7 for job to I/O page
                                ;
000670  032761  000000G 000000G         BIT     #$RNIOP,LSW9(R1);Is I/O page access wanted?
000676  001403                         BEQ     17$             ;Br if not
000700  052761  000000G 000000G         BIS     #$IOMAP,LSW6(R1);Set flag saying to map PAR 7 to I/O page
                                ;
                                ;  Make sure memory management is set up correctly for the job
                                ;
000706  004737  000000G         17$:    CALL    SETMAP          ;Set up memory mapping for the job
                                ;
                                ;  See if we need to lock this program in low memory
                                ;
000712  032761  000000G 000000G         BIT     #$RNMLK,LSW9(R1);Should we lock this program in memory?
000720  001410                         BEQ     18$             ;Br if not
000722  042761  000000G 000000G         BIC     #$RNMLK,LSW9(R1);Clear the flag
000730  012700  000000G                 MOV     #CSIARE,R0      ;Point to EMT arg block area
000734  012710  060007                 MOV     #<7+<140*400>>,(R0) ;Set up EMT argument block
000740  104375                         EMT     375             ;Try to lock job in memory
                                ;
                                ;  Give job a full time-slice
                                ;
000742  005061  000000G         18$:    CLR     LQUAN(R1)       ;GIVE FULL TIME-SLICE ON STARTUP
                                ;
                                ;  Push starting PSW and PC onto stack
                                ;
000746  012706  000000G                 MOV     #JSTK,SP        ;CLEAN OFF CONTEXT-BLOCK STACK
000752  012746  000000C                 MOV     #UMODE!UPMODE,-(SP);USER-MODE PS
000756  013746  000000G                 MOV     USTART,-(SP)    ;STARTING ADDRESS
                                ;
                                ;  See if we are to start program under the debugger
                                ;
000762  032761  000000G 000000G         BIT     #$DEBUG,LSW9(R1);Are we to run program under the debugger?
000770  001402                         BEQ     15$             ;Br if not
000772  000137  000000G                 JMP     DBGENT          ;Enter debugger to start the program
                                ;
                                ;  Use RTI to enter user's program in user mode.
                                ;
000776  000002                 15$:    RTI                     ;ENTER PROGRAM IN USER-MODE
                                ;
                                ;  Error reading SAV file
                                ;
001000  105037  000000G         9$:     CLRB    CINFLG          ;Say chain is not being done
001004                                  .CLOSE  #17             ;Close the SAV file channel
001012  012705  000007                 MOV     #EC7,R5         ;Get error code
001016  005004                         CLR     R4
001020  000137  000000G                 JMP     ABORT           ;Abort the job
```

```
                                ;-------------------------------------------------------------------------------
                                ;  Perform special initialization when starting execution of SETUP program.
                                ;  We must do this because SETUP tries to call the GETCSR routine to
                                ;  get the CSR address of the video controller when running on the PRO.
                                ;
001024                          SETINI:
                                ;
                                ;  Set up a special RMON pointer for SETUP which will cause offset
                                ;  434 to point to a routine which will return video CSR address.
                                ;
001024  012746  177344G                 MOV     #SUPRTN-434,-(SP);Set fake address for RMON pointer
001030  106637  0000006                 MTPD    @#RMON          ;Store RMON pointer into job space
001034  012700  0000000                 MOV     #SUPRTN,R0      ;Get location where we store code in job
001040  012746  0000020                 MOV     #SUPRTN+2,-(SP) ;Set address where code will be stored
001044  106620                          MTPD    (R0)+           ;Store into job space
                                ;
                                ;  Now store code in job space which will return CSR address for
                                ;  video controller on the top of the stack.
                                ;
001046  012746  012766                  MOV     #012766,-(SP)   ; [MOV  #csr,2(SP)]
001052  006620                          MTPI    (R0)+           ;Store into program space
001054  013746  0000006                 MOV     VIDCSR,-(SP)    ;Get CSR address for video controller
001060  006620                          MTPI    (R0)+           ;Store into program space
001062  012746  000002                  MOV     #2,-(SP)
001066  006620                          MTPI    (R0)+           ;Store stack offset for MOV instruction
001070  012746  000207                  MOV     #207,-(SP)      ; [RETURN]
001074  006620                          MTPI    (R0)+           ;Store into program space
                                ;
                                ;  Finished
                                ;
001076  000207                          RETURN
```

```
                                ;-------------------------------------------------------------------------------
                                ;  Move chain data from job context area to locations starting at 500
                                ;  in the memory image of the program being started.
                                ;
                                ;  Inputs:
                                ;   R4 = Number of words to move.
                                ;
001100  010346          CINMOV: MOV     R3,-(SP)
001102  010446                  MOV     R4,-(SP)
                                ;
                                ;  Move the data
                                ;
001104  012700  000500          MOV     #500,R0         ;Get address of area to receive data
001110  012703  0000000         MOV     #CINDAT,R3      ;Point to chain data area in context block
001114  012346          2$:     MOV     (R3)+,-(SP)     ;Move chain data from context block to job
001116  106620                  MTPD    (R0)+
001120  077403                  SOB     R4,2$
                                ;
                                ;  Finished
                                ;
001122  012604                  MOV     (SP)+,R4
001124  012603                  MOV     (SP)+,R3
001126  000207                  RETURN
```

```
                                ;-------------------------------------------------------------------------------
                                ;  KMRVM is called from KMRUN when setting up a job for execution.
                                ;  It checks to see if the job requires an extended-memory region for
                                ;  virtual overlays, and if so allocates the region and zeroes the
                                ;  segment id values in each section of the region.
                                ;
001130  010146                  KMRVM:  MOV     R1,-(SP)
001132  010246                          MOV     R2,-(SP)
001134  010346                          MOV     R3,-(SP)
001136  010446                          MOV     R4,-(SP)
001140  010546                          MOV     R5,-(SP)
                                ;
                                ;  Determine if this job needs an extended memory region
                                ;
001142  106537  000066                  MFPD    @#66            ;Get pointer to Window Definition Blocks
001146  022627  001000                  CMP     (SP)+,#1000     ;Any windows for job?
001152  103471                          BLO     9$              ;Br if not -- No XM region needed
                                ;
                                ;  Create an extended memory region using the information in the
                                ;  Region Definition Block in the job.
                                ;
001154  106537  000064                  MFPD    @#64            ;Get pointer to job's overlay table
001160  012601                          MOV     (SP)+,R1
001162  010102                          MOV     R1,R2           ;Point to region definition block
001164  062702  000000G                 ADD     #VO$RDB,R2
001170  106562  000000G                 MFPD    R.GSIZ(R2)      ;Get # 64-byte blocks to allocate for region
001174  012637  000000C                 MOV     (SP)+,RUNRDB+R.GSIZ
001200  001456                          BEQ     9$              ;Br if no space needed for XM region
001202  005037  000000C                 CLR     RUNRDB+R.GSTS   ;Clear Region Definition Block status flags
001206                                  .CRRG   #CSIARE,#RUNRDB ;Create the XM region
001226  103451                          BCS     10$             ;Br if error while creating the region
                                ;
                                ;  Zero the ID numbers in each section of the region.
                                ;  This is done to indicate that the code for the section needs to be
                                ;  read into the region from disk.
                                ;
001230  106561  000000G                 MFPD    VO$WDB(R1)      ;Get pointer to first Window Definition Block
001234  012602                          MOV     (SP)+,R2
001236  106561  000000G                 MFPD    VO$WDE(R1)      ;Get pointer past end of last WDB
001242  012603                          MOV     (SP)+,R3
001244  013704  000000C                 MOV     RUNRDB+R.GID,R4 ;Get address of system Region Control Block
001250  016400  000000G                 MOV     RC$BAS(R4),R0   ;Get base 64-byte page number of region base
001254  010446                  1$:     MOV     R4,-(SP)        ;Store pointer to Region Control Block in WDB
001256  106662  000000G                 MTPD    W.NRID(R2)
001262  010005                          MOV     R0,R5           ;Get base page number of region
001264  106562  000000G                 MFPD    W.NOFF(R2)      ;Get offset of window within region
001270  062605                          ADD     (SP)+,R5        ;Get base page # of window
001272                                  DISABL                  ;;; DISABLE
001300  013746  000000G                 MOV     @#KPAR0,-(SP)   ;;; Save mapping for kernel page 0
001304  010537  000000G                 MOV     R5,@#KPAR0      ;;; Map kernel page 0 to window section
001310  005037  000000                  CLR     @#0             ;;; Zero section ID number
001314  012637  000000G                 MOV     (SP)+,@#KPAR0   ;;; Restore system par 0 mapping
001320                                  ENABL                   ;;; ENABLE
001326  062702  000000G                 ADD     #W.SIZE,R2      ;Point to next window definition block
001332  020203                          CMP     R2,R3           ;Have we done all windows?
001334  103747                          BLO     1$              ;Loop if more windows to do
                                ;
```

```
                                        ;  Finished
                                        ;
001336  012605                  9$:       MOV     (SP)+,R5
001340  012604                            MOV     (SP)+,R4
001342  012603                            MOV     (SP)+,R3
001344  012602                            MOV     (SP)+,R2
001346  012601                            MOV     (SP)+,R1
001350  000207                            RETURN
                                        ;
                                        ;  Error -- Unable to create XM region for job
                                        ;
001352  012700  177754          10$:      MOV     #-24,R0         ;Return error code
001356  000137  0000000                   JMP     SETERR
```

```
                                        .SBTTL  Misc. Kmon EMTs
                                ;-----------------------------------------------------------------------
                                ;  SPOLGO (1) Initiate operation of a spooler.
                                ;  Argument word 2 contains the address of the SDCB for the spooled device
                                ;  being initiated.
                                ;
001362  105737  000000G        KMSPLG: TSTB    NSPLDV          ;Are there any spooled devices?
001366  001405                        BEQ     9$              ;BR IF NOT
001370  013700  000004G               MOV     EMTBLK+4,R0     ;GET ADDRESS OF SDCB
001374                                OCALL   SPOLGO          ;START THE SPOOLER
001402  000137  000000G        9$:     JMP     EMTXIT          ;FINISHED
```

```
                                ;-------------------------------------------------------------------------------
                                ;  Get data from within the system.
                                ;
                                ;  Argument words in EMT block:
                                ;   Word 2: Address of start of data within the kernel.
                                ;   Word 3: Number of bytes of data to transfer.
                                ;   Word 4: Address of buffer within TSKMON where data is to be placed.
                                ;
001406  013702  0000046G        KMSPEK: MOV     EMTBLK+4,R2     ;Get address of data wanted
001412  013703  0000060G                MOV     EMTBLK+6,R3     ;Get # bytes to transfer
001416  013704  0000100G                MOV     EMTBLK+10,R4    ;Get address of buffer in TSKMON
                                ;
                                ;  Move data from kernel to TSKMON buffer
                                ;
001422  006203                          ASR     R3              ;Get # words to transfer
001424  012246                  1$:     MOV     (R2)+,-(SP)     ;Get a word of data onto the stack
001426  106624                          MTPD    (R4)+           ;Store into TKMON buffer
001430  077303                          SOB     R3,1$           ;Loop till all words moved
                                ;
                                ;  Finished
                                ;
001432  000137  0000000G                JMP     EMTXIT          ;Finished
```

```
                                ;-----------------------------------------------------------------------------
                                ;  Store data into kernel data structure.
                                ;
                                ;  Argument words in EMT block:
                                ;   Word 2: Address of start of data within the kernel.
                                ;   Word 3: Number of bytes of data to transfer.
                                ;   Word 4: Address of buffer within TSKMON where data is to be fetched.
                                ;
001436  013702  000004G         KMSPOK: MOV     EMTBLK+4,R2     ;Get address of data destination
001442  013703  000006G                 MOV     EMTBLK+6,R3     ;Get # bytes to transfer
001446  013704  000010G                 MOV     EMTBLK+10,R4    ;Get address of buffer in TSKMON
                                ;
                                ;  Move data from TSKMON buffer to kernel
                                ;
001452  006203                  1$:     ASR     R3              ;Get # words to transfer
001454  106524                          MFPD    (R4)+           ;Get a word of data from TSKMON buffer
001456  012622                          MOV     (SP)+,(R2)+     ;Store into kernel space
001460  077303                          SOB     R3,1$           ;Loop till all words moved
                                ;
                                ;  Finished
                                ;
001462  000137  000000G                 JMP     EMTXIT          ;Finished
```

```
                                ;-----------------------------------------------------------------------
                                ;  Suspend or resume execution of a job.
                                ;
001466  013701  0000040         KMSPRS: MOV     EMTBLK+4,R1     ;Get # of job to suspend or resume
001472  001420                         BEQ     9$              ;Zero is invalid
001474  032761  0000000 0000000        BIT     #$DILUP,LSW(R1) ;Is job logged on?
001502  001414                         BEQ     9$              ;Br if not
001504  105737  0000000                TSTB    EMTBLK          ;Suspend or resume the job?
001510  001004                         BNE     1$              ;Br if resume
                                ;
                                ;  Suspend execution of the job
                                ;
001512  052761  0000000 0000000        BIS     #$SUSPN,LSW(R1) ;Set suspend-execution flag for job
001520  000405                         BR      9$
                                ;
                                ;  Resume execution of the job
                                ;
001522  042761  0000000 0000000 1$:     BIC     #$SUSPN,LSW(R1) ;Clear suspend-job flag
001530  004737  0000000                CALL    FORCEX          ;Force job to run
                                ;
                                ;  Finished
                                ;
001534  000137  0000000         9$:     JMP     EMTXIT
```

```
                                ;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                                ;  Cross connect a TT line to a CL unit.
                                ;  Word 3 of EMT argument block contains CL unit number.
                                ;
001540                          CLXCON:
                                ;
                                ;  Treat this as a NOP if this is a detached job
                                ;
001540  032761  0000000 0000000         BIT     #$DETCH,LSW(R1) ;Is this a detached job?
001546  001050                          BNE     9$              ;Br if yes
                                ;
                                ;  Ensure that pending terminal output is flushed
                                ;
001550  026161  0000000 0000000 2$:     CMP     LOTSPC(R1),LOTSIZ(R1)   ;Any output still pending?
001556  001405                          BEQ     3$              ;Go ahead if not
001560  012700  0000000                 MOV     #S$CPU,R0       ;Get out of everybody's way while waiting
001564  004737  0000000                 CALL    ENQTL           ;Give somebody else a chance while waiting
001570  000767                          BR      2$              ;And check again
                                ;
                                ;  Get Desired CL unit
                                ;
001572  013705  0000040         3$:     MOV     EMTBLK+4,R5     ;Get desired CL unit index number
                                ;
                                ;  Say CL unit connected to TT line
                                ;
001576  016104  0000000                 MOV     LNPRIM(R1),R4   ;Get primary line #
001602  010465  0000000                 MOV     R4,CL$XLN(R5)   ;Set # of TT line connected to CL
001606  042765  0000000 0000000         BIC     #^C<CM$BRK>,CL$STA(R5) ;Init status flags for CL unit
001614  052765  0000000 0000000         BIS     #CM$ON,CL$STA(R5) ;And say line is in use
                                ;
                                ;  Connect TT unit to CL line
                                ;
001622  010564  0000000                 MOV     R5,LXCL(R4)     ;Connect CL unit to our line
001626  012764  0000000 0000000         MOV     #CLXICP,LINIR(R4);Set addr of input char processing routine
                                ;
                                ;  Raise DTR on CL unit
                                ;
001634  052765  0000000 0000000         BIS     #CO$DTR,CL$OPT(R5) ;Request DTR up
001642                                  OCALL   SETDTR          ;Call CL routine to raise DTR
                                ;
                                ;  Suspend execution of job until cross connection is broken
                                ;
001650  005764  0000000         1$:     TST     LXCL(R4)        ;Are we still cross connected?
001654  002405                          BLT     9$              ;Br if not
001656  012700  0000000                 MOV     #S$SPND,R0      ;Get sleep state
001662  004737  0000000                 CALL    QNSPND          ;Suspend execution of job
001666  000770                          BR      1$              ;See if connection has been broken
                                ;
                                ;  Finished
                                ;
001670  000137  0000000         9$:     JMP     EMTXIT
```

```
                                        ;-----------------------------------------------------------------------
                                        ;  Start a terminal line
                                        ;
001674  013701  000004G         STTTLN: MOV     EMTBLK+4,R1     ;Get index # of line to be started
001700  032761  000000G 000000G        BIT     #$DILUP,LSW(R1) ;Is line already started?
001706  001017                         BNE     9$              ;Br if yes
001710  032761  000000G 000000G        BIT     #$AUTO,ILSW2(R1);Is line set to autobaud?
001716  001404                         BEQ     1$              ;Br if not
001720  012700  000000G                MOV     #S9600,R0       ;Set speed to 9600 baud
001724  004737  000000G                CALL    SETSPD          ;Set the line speed
001730  005000                 1$:     CLR     R0              ;No secondary start-up command file
001732                                 QCALL   INITLN          ;Initialize the line
001740  042761  000000G 000000G        BIC     #$RBRK,LSW10(R1);Clear break-received flag
001746  000137  000000G         9$:     JMP     EMTXIT
```

```
                                ;+------------------------------------------------------------------------------
                                ;  Force logoff of a job.
                                ;
001752  013702  000004G        KILEMT: MOV     EMTBLK+4,R2     ;GET # OF JOB TO KILL
001756  004737  006650'                CALL    CKACJB          ;See if we have privilege to kill that job
001762  103405                        BCS     10$             ;Br if don't have privilege
001764  010200                        MOV     R2,R0           ;Get # of job to kill
001766  004737  002004'                CALL    KILJOB          ;KILL THE JOB
001772  000137  000000G                JMP     EMTXIT          ;FINISHED
001776  005000                10$:    CLR     R0              ;Return error code 0
002000  000137  000000G                JMP     SETERR
                                ;+------------------------------------------------------------------------------
                                ;  Subroutine to "kill" (force logoff) of a job.
                                ;
                                ;  Inputs:
                                ;   R0 = Virtual job index number of job to kill.
                                ;
002004  010146                  KILJOB: MOV     R1,-(SP)
002006  010546                          MOV     R5,-(SP)
002010  010001                          MOV     R0,R1           ;Get virtual job index number
                                ;
                                ;  See if job is currently logged on
                                ;
002012  032761  000000G 000000G         BIT     #$DILUP,LSW(R1) ;IS JOB LOGGED ON NOW?
002020  001435                          BEQ     9$              ;BR IF NOT
                                ;
                                ;  If this line is cross connected with a CL line, break the connection.
                                ;
002022  016105  000000G                 MOV     LNPRIM(R1),R5   ;Get primary line number
002026  020527  000000G                 CMP     R5,#LSTPL       ;Is this a primary line (not detached)?
002032  101006                          BHI     1$              ;Br if detached job
002034  016505  000000G                 MOV     LXCL(R5),R5     ;Is line cross connected to a CL unit?
002040  002403                          BLT     1$              ;Br if not
002042                                  OCALL   CLXBRK          ;Break the connection
                                ;
                                ;  Clear ctrl-S output suspension and ctrl-O output suppression and
                                ;  set flag saying job is being killed.
                                ;
002050  016100  000000G         1$:     MOV     LNPRIM(R1),R0   ;GET PRIMARY LINE NUMBER
002054  042760  000000G 000000G         BIC     #$CTRLS,LSW3(R0);RESET CTRL-S OUTPUT SUSPENSION
002062  042761  000000G 000000G         BIC     #$CTRLO,LSW3(R1);RESET CTRL-O OUTPUT SUSPENSION
002070  042761  000000G 000000G         BIC     #$SUSPN,LSW(R1) ;Clear job-suspended flag
002076  052761  000000G 000000G         BIS     #$DISCN,LSW(R1) ;FORCE LOGOFF
                                ;
                                ;  Start output transmission for job
                                ;
002104  004777  000000G                 CALL    @TRNSTR         ;Start output transmission
                                ;
                                ;  Force job to execute
                                ;
002110  004737  000000G                 CALL    FORCEX          ;FORCE ITS EXECUTION
                                ;
                                ;  Finished
                                ;
002114  012605                  9$:     MOV     (SP)+,R5
002116  012601                          MOV     (SP)+,R1
```

```
     58 002120  000207                          RETURN
```

```
                                ;-----------------------------------------------------------------------
                                ;  Clean out data cache.
                                ;
002122  005737  000000G         CSHNEW: TST     CSHALC          ;Is data caching genned into system?
002126  001402                          BEQ     1$              ;Br if not
002130  004777  000000G                 CALL    @CSHINI         ;Clean out the data cache
002134  000137  000000G         1$:     JMP     EMTXIT
```

```
                                ;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                                ;  Delete a spool file.
                                ;   Form of the EMT:
                                ;        .BYTE   0,126
                                ;        .WORD   7
                                ;        .WORD   file_id_number
                                ;
002140                          KMSPDL:
                                ;
                                ;  See if spooling support is included in this system
                                ;
002140  105737  0000006                 TSTB    NSPLDV          ;Are there any spooled devices?
002144  001410                          BEQ     1$              ;Br if not
                                ;
                                ;  Get the file id number and delete the file
                                ;
002146  013703  0000046                 MOV     EMTBLK+4,R3     ;Get spool file ID number
002152                                  OCALL   SPLDEL          ;Try to delete the file
002160  103402                          BCS     1$              ;Br if could not find the file
002162  000137  0000006                 JMP     EMTXIT
                                ;
                                ;  Specified file could not be found
                                ;
002166  012700  000001          1$:     MOV     #EC1,R0
002172  000137  0000000                 JMP     SETERR
```

```
                                        .SBTTL  VLSEMT -- EMT to switch between subprocesses
                                ;-----------------------------------------------------------------
                                ;  The form of the EMT used to switch between subprocesses is
                                ;
                                ;       .BYTE   sub_function,162
                                ;       .BYTE   subprocess_number,0
                                ;       .BYTE   return_process,initiate_only
                                ;       .WORD   pointer_to_start_up_file_string
                                ;
                                ;  Where subprocess_number is the relative subprocess number for the
                                ;  job (0=primary process).
                                ;
002176  005037  000000G       VLSEMT: CLR     @R0             ;Return 0 in R0 if error
                                ;
                                ;  See if job is authorized to use subprocesses
                                ;
002202  032761  000000G 000000G        BIT     #$NOVLN,LSW2(R1);Is job allowed to use subprocesses?
002210  001014                        BNE     1$              ;Br if not
002212  032761  000000G 000000G        BIT     #$DETCH,LSW(R1) ;Is this a detached job?
002220  001010                        BNE     1$              ;Br if detached
002222  032761  000000C 000000G        BIT     #<$SUCF!$LOFCF>,LSW9(R1) ;Doing logon/logoff processing?
002230  001004                        BNE     1$              ;Br if yes
002232  032761  000000G 000000G        BIT     #$NOIN,LSW3(R1) ;Doing logon/logoff processing?
002240  001404                        BEQ     2$              ;Br if not
002242  012700  000001          1$:   MOV     #EC1,R0         ;Error 1 if not allowed to use subprocesses
002246  000137  000000G               JMP     SETERR
                                ;
                                ;  See if he specified a process number or wants us to find a free one
                                ;
002252  113705  000002G         2$:   MOVB    EMTBLK+2,R5     ;Get specified subprocess number
002256  001570                        BEQ     4$              ;Br if switching to primary process
002260  003015                        BGT     12$             ;Br if subprocess number specified
002262  012705  000001                MOV     #1,R5           ;Start with subprocess # 1
002266  016103  000000G               MOV     LNPRIM(R1),R3   ;Get primary line #
002272  016303  000000G               MOV     LSECPT(R3),R3   ;Point to table of secondary line numbers
002276  020527  000000G         13$:  CMP     R5,#MAXSEC      ;Are there more subprocesses available to job?
002302  101050                        BHI     6$              ;Br if not
002304  105723                        TSTB    (R3)+           ;Is this subprocess free?
002306  001423                        BEQ     15$             ;Br if free
002310  005205                        INC     R5              ;Advance subprocess number
002312  000771                        BR      13$             ;Continue searching for a free subprocess
                                ;
                                ;  See if the specified process number is valid
                                ;
002314  020527  000000G         12$:  CMP     R5,#MAXSEC      ;Is subprocess number too large?
002320  101407                        BLOS    3$              ;Br if ok
002322  012737  000000G 000000G        MOV     #MAXSEC,@R0     ;Return MAXSEC to user in R0
002330  012700  000002                MOV     #EC2,R0         ;Error 2 -- Subprocess number too big
002334  000137  000000G               JMP     SETERR
                                ;
                                ;  Switching to a subprocess.
                                ;  See if subprocess has been allocated yet.
                                ;
002340  016102  000000G         3$:   MOV     LNPRIM(R1),R2   ;Get primary line #
002344  016203  000000G               MOV     LSECPT(R2),R3   ;Point to table of secondary line numbers
002350  060503                        ADD     R5,R3           ;Point to entry for selected process
002352  105743                        TSTB    -(R3)           ;Has this subprocess been allocated yet?
```

```
     58 002354  001131                          BNE     4$              ;Br if yes
     59                                 ;
     60                                 ;  Request to switch to a subprocess which has not been allocated yet.
     61                                 ;
     62 002356  123727  000000G 000002  15$:    CMPB    EMTBLK,#2       ;Is it ok to start a new subprocess?
     63 002364  001004                          BNE     18$             ;Br if yes
     64 002366  012700  000006                  MOV     #EC6,R0         ;Error 6 -- Subprocess not active
     65 002372  000137  000000G                 JMP     SETERR
     66                                 ;
     67                                 ;  See if we can find a free subprocess.
     68                                 ;
     69 002376  012703  000000G         18$:    MOV     #FSTSL,R3       ;Get job # of first subprocess
     70 002402  020327  000000G         5$:     CMP     R3,#LSTSL       ;Checked all?
     71 002406  101006                          BHI     6$              ;Br if yes
     72 002410  005763  000000G                 TST     LNPRIM(R3)      ;Is this line available?
     73 002414  001407                          BEQ     7$              ;Br if yes
     74 002416  062703  000002                  ADD     #2,R3           ;Get index to next subprocess
     75 002422  000767                          BR      5$              ;Continue search
     76                                 ;
     77                                 ;  There are no free subprocesses
     78                                 ;
     79 002424  012700  000003          6$:     MOV     #EC3,R0         ;Error 3 -- No free subprocesses
     80 002430  000137  000000G                 JMP     SETERR
     81                                 ;
     82                                 ;  We have found a free subprocess.
     83                                 ;  Its index is in R3.
     84                                 ;  Validate address of pointer to secondary start-up command file name string.
     85                                 ;
     86 002434  013700  000006G         7$:     MOV     EMTBLK+6,R0     ;Get virtual address of name string
     87 002440  001402                          BEQ     17$             ;Br if no start-up file specified
     88 002442  004737  000000G                 CALL    VALADB          ;Validate the address
     89                                 ;
     90                                 ;  Allocate an I/O queue element to use to pass information to the
     91                                 ;  job initiator.
     92                                 ;
     93 002446  010146                  17$:    MOV     R1,-(SP)        ;Save our job index number
     94 002450  010346                          MOV     R3,-(SP)
     95 002452  004737  000000G                 CALL    GETQ            ;Get a free I/O queue element (R1=address)
     96 002456  105061  000000G                 CLRB    IB$SF2(R1)      ;Assume no secondary start-up command file
     97                                 ;
     98                                 ;  See if we need to set up a secondary start-up command file for the
     99                                 ;  process that is being initiated.
    100                                 ;
    101 002462  013703  000006G                 MOV     EMTBLK+6,R3     ;Is there a start-up command file string?
    102 002466  001412                          BEQ     16$             ;Br if no secondary start-up command file
    103 002470  010104                          MOV     R1,R4           ;Point to buffer where name will be stored
    104 002472  062704  000000G                 ADD     #IB$SF2,R4      ;Point to cell in buffer for name
    105 002476  012702  177777G                 MOV     #SF2LEN-1,R2    ;Get # chars reserved for file name
    106 002502  004737  000000G         11$:    CALL    GETUCH          ;Get char from user's buffer
    107 002506  110024                          MOVB    R0,(R4)+        ;Store into buffer
    108 002510  077204                          SOB     R2,11$          ;Loop till all of name moved
    109 002512  105014                          CLRB    (R4)            ;Make sure there is a null at the end
    110                                 ;
    111                                 ;  Complete initialization of buffer passing info to job initiator
    112                                 ;
    113 002514  010100                  16$:    MOV     R1,R0           ;Save pointer to initiation info buffer
    114 002516  012603                          MOV     (SP)+,R3
```

```
                                        MOV     (SP)+,R1        ;Get back current process index
                                        TSTB    EMTBLK+4        ;Return to our process or primary process?
                                        BEQ     10$             ;Br if return to primary process
                                        MOVB    R1,IB$IJ(R0)    ;Set our process # as initiating job
                                ;
                                ;  At this point, R0=Pointer to buffer with initialization info (IB$xxx).
                                ;  Initialize the subprocess.
                                ;
                                10$:    MOV     R1,-(SP)        ;Save original process number
                                        MOV     R3,R1           ;Get new process number
                                        OCALL   INITLN          ;Initialize the new process
                                        BCC     8$              ;Br if initialization ok
                                        MOV     #EC4,R0         ;Error 4 -- Insufficient memory for process
                                        JMP     SETERR
                                8$:     BIS     #$VBELL,LSW9(R1);Don't ring bell when new subprocess waits
                                        TSTB    EMTBLK+5        ;Are we going to switch terminal control?
                                        BEQ     20$             ;Br if yes
                                        BIS     #$VNOTT,LSW(R1) ;Say subprocess not connected to terminal
                                        OCALL   SPPRED          ;Reduce execution priority of initiated proc
                                20$:    MOV     (SP)+,R1        ;Get back original process number
                                ;
                                ;  Make association between subprocess and primary process
                                ;
                                        MOV     LNPRIM(R1),R2   ;Get primary process number
                                        MOV     R2,LPARNT(R3)   ;Say primary job is parent of subprocess
                                        MOV     LSECPT(R2),R4   ;Point to subprocess # table
                                        ADD     R5,R4           ;Point to desired entry
                                        MOVB    R3,-(R4)        ;Associate subprocess with primary
                                        MOV     R2,LNPRIM(R3)   ;Set primary process for subprocess
                                        BR      19$
                                ;
                                ;  We are switching to a process that has been initialized.
                                ;
                                4$:     CMPB    EMTBLK,#1       ;Can we use an existing process?
                                        BNE     19$             ;Br if yes
                                        MOV     #EC5,R0         ;Error 5 -- Process already exists
                                        JMP     SETERR
                                ;
                                ;  Switch terminal connection to the selected process.
                                ;
                                19$:    TSTB    EMTBLK+5        ;Initiate subprocess only?
                                        BNE     9$              ;Br if yes -- Don't switch terminal control
                                        OCALL   LINSWT          ;Set up line number prompt
                                ;
                                ;  Set flag to suppress bell ringing when this line goes into wait state
                                ;
                                        BIS     #$VBELL,LSW9(R1);Say not to ring bell when we wait
                                ;
                                ;  Return subprocess number to calling program in R0
                                ;
                                9$:     MOV     R5,UR0          ;Return subprocess # in R0
                                ;
                                ;  Finished
                                ;
                                        JMP     EMTXIT          ;Finished with EMT
```

```
                        .SBTTL  Detached job control emt's
                ;-----------------------------------------------------------------------
                ;  This set of emt's is used to control detached jobs.
                ;  The emt function code is 132.  The sub-function code is stored in
                ;  byte 0 of the argument block.  The valid sub-function codes are:
                ;   0==>Start a detached job.
                ;   1==>Check status of a detached job.
                ;   2==>Kill a detached job.
                ;
002712          DETEMT:
                ;
                ;  Jump to processing routine based on sub-function code.
                ;
002712  113700  000000G         MOVB    EMTBLK,R0       ;GET SUB-FUNCTION CODE
002716  020027  000003          CMP     R0,#DETHI       ;IS SUB-FUNCTION VALID?
002722  103003                  BHIS    2$              ;BR IF NOT
002724  006300                  ASL     R0              ;CONVERT TO WORD TABLE INDEX
002726  000170  003226'          JMP     @DETVEC(R0)     ;ENTER PROCESSING ROUTINE
002732  005000          2$:     CLR     R0              ;Return error code 0
002734  000137  000000G         JMP     SETERR
                ;
                ;  Start a new detached job.
                ;
002740  032737  000000G 000000G DETGO:  BIT     #PO$DET,PRIVCO  ;Is this user allowed to access detached jobs?
002746  001003                  BNE     10$             ;Br if yes
002750  005000                  CLR     R0              ;Return error code 0 if not
002752  000137  000000G         JMP     SETERR
                ;
                ;  Search for a free detached job line.
                ;
002756  012701  000000G 10$:     MOV     #FSTDL,R1       ;GET # OF FIRST DETACHED JOB LINE
002762  020127  000000G 3$:      CMP     R1,#LSTDL       ;HAVE WE CHECKED ALL?
002766  101043                  BHI     DETONE          ;BR IF YES -- NONE FREE
002770  032761  000000G 000000G BIT     #$DILUP,LSW(R1) ;IS THIS LINE FREE?
002776  001403                  BEQ     2$              ;BR IF YES
003000  062701  000002          ADD     #2,R1           ;GO TRY NEXT
003004  000766                  BR      3$
                ;  Found a free detached job line.
                ;  Set name of start-up command file.
003006  013703  000020G 2$:      MOV     EMTBLK+2,R3     ;GET ADDRESS OF START-UP FILE NAME ARGUMENT
003012  010300                  MOV     R3,R0           ;VALIDATE THE ADDRESS
003014  004737  000000G         CALL    VALADB
003020  016102  000000G         MOV     LSUCF(R1),R2    ;GET ADDRESS WHERE WE SHOULD STORE NAME
003024  012705  000000G         MOV     #DETCBS,R5      ;GET COUNT OF MAX # CHARS TO MOVE
003030  004737  000000G 4$:      CALL    GETUCH          ;GET CHAR FROM USER'S BUFFER
003034  110022                  MOVB    R0,(R2)+        ;STORE AS START-UP FILE NAME FOR LINE
003036  001402                  BEQ     6$              ;BR IF MOVED NULL AT END OF NAME
003040  077505                  SOB     R5,4$           ;LOOP TO MOVE REST OF NAME
003042  105012                  CLRB    (R2)            ;PUT NULL AT END OF NAME
                ;  Start up the line.
003044  005000          6$:     CLR     R0              ;No secondary start-up command file
003046                  OCALL   INITLN          ;INITIALIZE THE LINE
003054  103410                  BCS     DETONE          ;BR IF INSUFFICIENT MEMORY TO START JOB
                ;  Set number of parent job
003056  113761  000000G 000000G MOVB    CORUSR,LPARNT(R1);Set # of parent job
                ;  Return detached job line number to user in R0.
003064  006201                  ASR     R1              ;CONVERT TO LINE NUMBER
```

```
     58 003066  010137  000000G                MOV     R1,UR0          ;RETURN IN USER'S R0
     59 003072  000137  000000G                JMP     EMTXIT          ;FINISHED
     60                                ;  No free detached job lines -- Return error code 1.
     61 003076  012700  000001         DETONE: MOV     #EC1,R0
     62 003102  000137  000000G         DETERR: JMP     SETERR
     63                                ;
     64                                ;  Check status of a detached job.
     65                                ;
     66 003106  013701  000020         DETCHK: MOV     EMTBLK+2,R1     ;GET LINE NUMBER
     67 003112  006301                         ASL     R1              ;CONVERT TO LINE INDEX
     68 003114  020127  000000G                CMP     R1,#LSTDL       ;IS THIS A VALID DETACHED JOB LINE #?
     69 003120  101366                         BHI     DETONE          ;BR IF NOT
     70 003122  032761  000000G 000000G       BIT     #$DILUP,LSW(R1) ;IS JOB STILL ACTIVE?
     71 003130  001762                         BEQ     DETONE          ;BR IF NOT
     72 003132  000137  000000G                JMP     EMTXIT          ;NO ERROR IF STILL ACTIVE
     73                                ;
     74                                ;  Kill a detached job
     75                                ;
     76 003136  013702  000020         DETKIL: MOV     EMTBLK+2,R2     ;GET JOB NUMBER
     77 003142  006302                         ASL     R2              ;CONVERT TO LINE INDEX NUMBER
     78 003144  020227  000000G                CMP     R2,#LSTPL       ;PRIMARY LINE?
     79 003150  103416                         BLO     3$              ;IF SO, DON'T CHECK DET PRIV
     80 003152  020227  000000G                CMP     R2,#LSTDL       ;VALID DETACHED JOB #?
     81 003156  101010                         BHI     2$              ;IF NOT, DON'T CHECK DET PRIV
     82 003160  032737  000000G 000000G       BIT     #PO$DET,PRIVCO  ;DO WE HAVE DETACHED PRIV?
     83 003166  001007                         BNE     3$              ;IF WE HAVE PRIV, CONTINUE
     84 003170  012700  000002         4$:     MOV     #EC2,R0         ;ELSE, SAY NOT PRIVLIGED
     85 003174  000137  000000G                JMP     SETERR          ;AND ABORT
     86 003200  020227  000000G         2$:     CMP     R2,#LSTSL       ;VIRTUAL(SECONDARY) LINE?
     87 003204  101334                         BHI     DETONE          ;BR IF NOT (INVALID LINE #)
     88 003206  004737  006650'        3$:     CALL    CKACJB          ;Can we access that job?
     89 003212  103766                         BCS     4$              ;If not, report priv error
     90 003214  010200                 1$:     MOV     R2,R0           ;Get job index number
     91 003216  004737  002004'                CALL    KILJOB          ;KILL THE JOB
     92 003222  000137  000000G                JMP     EMTXIT          ;FINISHED
     93                                ;
     94                                ;  Detached job jump vector
     95                                ;
     96 003226  002740'                DETVEC: .WORD   DETGO           ; 0 - Start a detached job
     97 003230  003106'                        .WORD   DETCHK          ; 1 - Check status of a detached job
     98 003232  003136'                        .WORD   DETKIL          ; 2 - Kill a detached job
     99         000003                 DETHI   =       <.-DETVEC>/2
```

```
                                        .SBTTL  Shared Run-time control emt's
                                ;-------------------------------------------------------------------
                                ;  All of the shared run-time control emt's have an emt function
                                ;  code of 143.  The sub-function code is stored in byte 0 of the
                                ;  emt argument block.
                                ;
                                ;  Sub-functions:
                                ;   0 ==> Associate or disassociate a run-time system.
                                ;   1 ==> Map part of a run-time system into job's virtual memory area.
                                ;   2 ==> Define shared run-time region for fast mapping.
                                ;   3 ==> Disable fast run-time region mapping (restore TRAP instr.)
                                ;   4 ==> Enable I/D space split
                                ;   5 ==> Disable I/D space split (return to all I-space)
                                ;   6 ==> Define PLAS region for fast mapping
                                ;
003234  113700  0000000         SSEMT:  MOVB    EMTBLK,R0       ;Get sub-function code
003240  020027  000007                  CMP     R0,#SSEHI       ;Valid sub-function code?
003244  103402                          BLO     1$              ;Br if OK
003246  000137  0000000                 JMP     BADEMT          ;Invalid sub-function
003252  006300                  1$:     ASL     R0              ;Convert sub-function to word index
003254  000170  003260'                 JMP     @SSEVEC(R0)     ;Jump to appropriate routine
                                ;
                                ;  Defind jump vector table for mapping emts
                                ;
003260  003276'                 SSEVEC: .WORD   SSASOC          ;0 - associate a run-time system
003262  003450'                         .WORD   SSMAP           ;1 - map to shared run-time system
003264  003574'                         .WORD   SSREGN          ;2 - define fast map to shared run-time
003266  003774'                         .WORD   SSNOMP          ;3 - disable fast mapping (restore TRAP use)
003270  004132'                         .WORD   ESPLIT          ;4 - enable I/D space split
003272  004172'                         .WORD   DSPLIT          ;5 - restore all I space
003274  004220'                         .WORD   SSPLAS          ;6 - define fast map to PLAS region
                                ;
        000007                  SSEHI   = <.-SSEVEC>/2          ;This and higher subfunctions are invalid
                                ;
```

```
                                        .SBTTL  Associate shared run-time system with job
                                ;-------------------------------------------------------------------
                                ;  This emt is used to associate a shared run-time system with a job.
                                ;  The form of the emt argument block is:
                                ;
                                ;       .BYTE   0,143
                                ;       .WORD   name-pointer
                                ;
                                ;  Where name-pointer is the address of a two word cell containing the
                                ;  rad50 name of the run-time system to be associated.
                                ;
003276  013700  0000020       SSASOC: MOV     EMTBLK+2,R0     ;GET ADDRESS OF NAME CELL
003302  001026                        BNE     2$              ;BR IF WE ARE ASSOCIATING A RUN-TIME
                                ;
                                ;  Name pointer is zero -- Disassociate all run-time systems
                                ;
003304  005037  0000000               CLR     CURRDB          ;NO ASSOCIATED RUN-TIME
003310  005002                        CLR     R2              ;INIT PAR INDEX
003312  005062  0000000       1$:     CLR     RPAR(R2)        ;CLEAR PAR VALUE
003316  005062  0000000               CLR     RPDR(R2)        ;CLEAR PDR VALUE
003322  062702  000002                ADD     #2,R2           ;ADVANCE TO NEXT SET
003326  020227  000016                CMP     R2,#2*7         ;DONE ALL?
003332  101767                        BLOS    1$              ;BR IF MORE TO CLEAR
003334  005737  0000000               TST     VPLAS           ;Do we have PLAS support?
003340  001403                        BEQ     7$              ;Br if not
003342                                OCALL   MAPPLS          ;Set up info about any PLAS regions
003350  004737  0000000       7$:     CALL    SETMAP          ;RELOAD MAP REGISTERS FOR JOB
003354  000137  0000000               JMP     EMTXIT          ;FINISHED
                                ;
                                ;  Associate new shared run-time system with job
                                ;
003360  004737  0000000       2$:     CALL    VALADW          ;VALIDATE THE ADDRESS
003364  106520                        MFPD    (R0)+           ;GET 1ST WORD OF RUN-TIME NAME
003366  012604                        MOV     (SP)+,R4
003370  106520                        MFPD    (R0)+           ;GET 2ND WORD OF RUN-TIME NAME
003372  012605                        MOV     (SP)+,R5
                                ;
                                ;  Search for run-time system descriptor with matching name
                                ;
003374  012703  0000000               MOV     #RDB,R3         ;POINT TO 1ST RUN-TIME DESCRIPTOR BLOCK
003400  020327  0000000       3$:     CMP     R3,#RDBEND      ;CHECKED ALL RUN-TIME DESCRIPTOR BLOCKS?
003404  103015                        BHIS    6$              ;BR IF YES
003406  020463  0000000               CMP     R4,RT$NAM(R3)   ;COMPARE 1ST WORD OF NAME
003412  001003                        BNE     5$              ;BR IF MISMATCH
003414  020563  0000020               CMP     R5,RT$NAM+2(R3) ;COMPARE 2ND WORD OF NAME
003420  001403                        BEQ     4$              ;BR IF FOUND RIGHT BLOCK
003422  062703  0000000       5$:     ADD     #RT$$SZ,R3      ;POINT TO NEXT RUN-TIME DESCRIPTOR BLOCK
003426  000764                        BR      3$              ;GO CHECK NEXT ONE
                                ;
                                ;  Found run-time descriptor block
                                ;  Associate the run-time system with the job
                                ;
003430  010337  0000000       4$:     MOV     R3,CURRDB       ;SAY RUN-TIME SYSTEM IS ASSOCIATED WITH JOB
003434  000137  0000000               JMP     EMTXIT          ;FINISHED
                                ;
                                ;  Error -- Cannot find named run-time system
                                ;  Return error code of 1
```

```
     58                                      ;
     59 003440  012700  000001          6$:  MOV     #EC1,R0          ;RETURN ERROR CODE 1
     60 003444  000137  000000G             JMP     SETERR
```

```
                                        .SBTTL  Map shared run-time system into user's virtual address
                                ;-------------------------------------------------------------------------
                                ;  This emt is used to map a portion of a shared run-time system into
                                ;  the virtual address region of the job.
                                ;  The form of the emt is:
                                ;
                                ;       .BYTE   1,143
                                ;       .WORD   user-par        ;0 to 7
                                ;       .WORD   run-time-offset ;64-byte units
                                ;       .WORD   size            ;64-byte units
                                ;
003450                          SSMAP:
                                ;
                                ;  Call routine to check arguments and bias base address
                                ;
003450  004737  004004'                 CALL    SSSTUP          ;Do common mapping setup
                                ;
                                ;  Set up mapping information for each par
                                ;
003454  010500                  3$:     MOV     R5,R0           ;CALCULATE # 64-BYTE BLOCKS
003456  160400                          SUB     R4,R0           ; LEFT TO ALLOCATE
003460  003443                          BLE     6$              ;BR IF FINISHED
003462  010462  0000000                 MOV     R4,RPAR(R2)     ;SET PAR BASE OFFSET VALUE
003466  010462  0000000                 MOV     R4,CUPAR0(R2)   ;SAVE INFORMATION IN JOB CONTEXT BLOCK
003472  010462  0000000                 MOV     R4,UPAR0(R2)    ;SET HARDWARE REGISTER
003476  020027  000200                  CMP     R0,#200         ;EACH PAR CAN MAP UP TO 200 64-BYTE BLOCKS
003502  101402                          BLOS    4$              ;BR IF WE CAN MAP ALL THAT'S LEFT
003504  012700  000200                  MOV     #200,R0         ;ONLY MAP 200 THROUGH THIS PAR
003510  060004                  4$:     ADD     R0,R4           ;ADVANCE BASE OFFSET NUMBER
003512  005300                          DEC     R0              ;PDR VALUE IS ACTUAL-1
003514  000300                          SWAB    R0              ;POSITION LENGTH FOR PDR REGISTER
003516  052700  000002                  BIS     #2,R0           ;SET READ-ALLOWED FLAG FOR PDR
003522  132763  0000000 0000000         BITB    #RF$WRT,RT$FLG(R3);IS WRITE ACCESS TO BE ALLOWED?
003530  001402                          BEQ     5$              ;BR IF NOT
003532  052700  000004                  BIS     #4,R0           ;SET WRITE-ALLOWED FLAG FOR PDR
003536  042700  100261          5$:     BIC     #100261,R0      ;MAKE SURE UNUSED PDR BITS ARE ZERO
003542  010062  0000000                 MOV     R0,RPDR(R2)     ;THIS IS PDR VALUE FOR PAR RANGE
003546  010062  0000000                 MOV     R0,CUPDR0(R2)   ;SAVE INFORMATION IN JOB CONTEXT BLOCK
003552  010062  0000000                 MOV     R0,UPDR0(R2)    ;SET HARDWARE MAPPING
003556  062702  000002                  ADD     #2,R2           ;ADVANCE PAR INDEX #
003562  020227  000016                  CMP     R2,#2*7         ;ONLY DO PAR UP TO # 7
003566  101732                          BLOS    3$              ;BR IF MORE TO DO
                                ;
                                ;  Finished
                                ;
003570  000137  0000000         6$:     JMP     EMTXIT          ;FINISHED
```

```
                                        .SBTTL  Define a run-time region for fast mapping
                                ;-------------------------------------------------------------------------
                                ;  Define a run-time region so that it can be quickly mapped using
                                ;  the TRAP instruction.
                                ;
                                ;  EMT argument list:
                                ;       .BYTE   2,143
                                ;       .WORD   user-par        ;PAR to map to region (0 to 7)
                                ;       .WORD   run-time-offset ;Offset within run-time (64-byte units)
                                ;       .WORD   size            ;# 64-byte units to map
                                ;       .WORD   region-index    ;0 to MAXSRD-1
                                ;
003574                          SSREGN:
                                ;
                                ;  Call routine to check arguments and bias base address
                                ;
003574  004737  004004'                 CALL    SSSTUP          ;Do common mapping setup
                                ;
                                ;  At this point the following registers are set up:
                                ;       R2 = PAR index number (2 * PAR).
                                ;       R3 = Pointer to current run-time descriptor block.
                                ;       R4 = Starting 64-byte physical address of region.
                                ;       R5 = Ending 64-byte physical address of region.
                                ;
                                ;  Make sure the region-index value is valid.
                                ;
003600  160405                          SUB     R4,R5           ;R5 now has # blocks to be mapped
003602  013700  000010G                 MOV     EMTBLK+10,R0    ;Get specified region #
003606  020027  000000G         1$:     CMP     R0,#MAXSRD      ;Is the number valid?
003612  103404                          BLO     11$             ;Br if ok
003614  012700  000002                  MOV     #EC2,R0         ;Return error code 2
003620  000137  000000G                 JMP     SETERR
                                ;
                                ;  Remember which PAR is set by this region
                                ;
003624  110260  000000G         11$:    MOVB    R2,SR$PX(R0)    ;Save PAR index (2*PAR #)
003630  105060  000000G                 CLRB    SR$FLG(R0)      ;No special mapping flags yet
                                ;
                                ;  Set the PAR value in the region descriptor
                                ;
003634  006300                          ASL     R0              ;Get word table index
003636  005060  000000G                 CLR     SR$WCB(R0)      ;This is a shared run-time region (not PLAS)
003642  010460  000000G                 MOV     R4,SR$PAR(R0)   ;Set base PAR value for region
                                ;
                                ;  Set the PDR value for the region
                                ;
003646  010501                          MOV     R5,R1           ;Get # blocks remaining to map
003650  020127  000200                  CMP     R1,#200         ;Will it all fit thru 1 par?
003654  101402                          BLOS    2$              ;Br if ok
003656  012701  000200                  MOV     #200,R1         ;If not, decrease to 1 par worth
003662  160105                  2$:     SUB     R1,R5           ;This many left to map next time
003664  060104                          ADD     R1,R4           ;Bump up base address for next time
003666  005301                          DEC     R1              ;PDR size is 1 less than wanted
003670  000301                          SWAB    R1              ;Position size for PDR value
003672  132763  000000G 000000G         BITB    #RF$WRT,RT$FLG(R3);Can we write to shared run-time?
003700  001402                          BEQ     3$              ;Br if not
003702  052701  000004                  BIS     #4,R1           ;Set flag in PDR value saying write ok
```

Define a run-time region for fast mapping

```
     58 003706  052701  000002          3$:     BIS     #2,R1           ;Set flag in PDR value saying read ok
     59 003712  042701  100261                  BIC     #100261,R1      ;Make sure unused bits are clear
     60 003716  010160  000000G                 MOV     R1,SR$PDR(R0)   ;Store PDR value
     61 003722  005705                          TST     R5              ;Any more left to map?
     62 003724  001416                          BEQ     5$              ;Br if not
     63                                 ;
     64                                 ;  More than 8Kb space requested, use more pars
     65                                 ;
     66 003726  062702  000002                  ADD     #2,R2           ;Increment to next par index
     67 003732  020227  000016                  CMP     R2,#2*7         ;Valid PAR index?
     68 003736  101006                          BHI     37$             ;Br if not
     69 003740  006200                          ASR     R0              ;Convert trap region index back to #
     70 003742  152760  000000G 000000G        BISB    #SR.MOR,SR$FLG(R0) ;Say there are more to map
     71 003750  005200                          INC     R0              ;Step up to next fast map region #
     72 003752  000715                          BR      1$              ;Repeat until entire window set up
     73                                 ;
     74                                 ;  Attempt to map past par 7
     75                                 ;
     76 003754  005000                  37$:    CLR     R0              ;Return ECO
     77 003756  000137  000000G                 JMP     SETERR          ;Return error to user
     78                                 ;
     79                                 ;  Finished
     80                                 ;
     81 003762  112737  000001  000000G 5$:     MOVB    #1,DOTRMP       ;Enable TRAP to do mapping
     82 003770  000137  000000G                 JMP     EMTXIT
     83
     84                                 ;-----------------------------------------------------------------------
     85                                 ;  Disable fast run-time mapping and return the TRAP instruction to
     86                                 ;  its standard use.
     87                                 ;
     88 003774  105037  000000G         SSNOMP: CLRB    DOTRMP          ;Say mapping not in effect
     89 004000  000137  000000G                 JMP     EMTXIT
```

```
                                ;+-----------------------------------------------------------------------
                                ;  Common routine used to process arguments for shared run-time
                                ;  mapping EMT's.
                                ;
                                ;  Outputs:
                                ;       R2 = PAR index number (2 * PAR).
                                ;       R3 = Pointer to current run-time descriptor block.
                                ;       R4 = Starting 64-byte physical address of region.
                                ;       R5 = Ending 64-byte physical address of region.
                                ;       UR0= Size of area that is being mapped.
                                ;
004004                          SSSTUP:
                                ;
                                ;  Check the specified PAR number
                                ;
004004  013702  000002G                 MOV     EMTBLK+2,R2     ;Get PAR #
004010  020227  000007                  CMP     R2,#7           ;Must be in range 0 to 7
004014  101404                          BLOS    7$              ;Br if ok
004016  012700  177767                  MOV     #-11,R0         ;Fatal error if other value
004022  000137  000000G                 JMP     SETERR
004026  006302                  7$:     ASL     R2              ;Convert par # to word table index
                                ;
                                ;  Get pointer to descriptor for current shared run-time system
                                ;
004030  013703  000000G                 MOV     CURRDB,R3       ;Get address of associated run-time descriptor
004034  001004                          BNE     1$              ;Better be one
004036  012700  000001                  MOV     #EC1,R0         ;If not then return error code 1
004042  000137  000000G                 JMP     SETERR
                                ;
                                ;  Get physical memory address
                                ;
004046  013704  000004G         1$:     MOV     EMTBLK+4,R4     ;Get run-time base offset
004052  020463  000000G                 CMP     R4,RT$TOP(R3)   ;Make sure it is within run-time area
004056  103404                          BLO     3$              ;Br if ok
004060  012700  000005                  MOV     #EC5,R0         ;Return error 5 if too high
004064  000137  000000G                 JMP     SETERR
004070  013705  000006G         3$:     MOV     EMTBLK+6,R5     ;Get size of region to be mapped
004074  060405                          ADD     R4,R5           ;Get offset above top of region to be mapped
004076  066304  000000G                 ADD     RT$BAS(R3),R4   ;Bias the offsets
004102  066305  000000G                 ADD     RT$BAS(R3),R5
                                ;
                                ;  Get amt of memory to be mapped
                                ;
004106  020563  000000G                 CMP     R5,RT$TOP(R3)   ;Is top of request above top of run-time?
004112  101402                          BLOS    2$              ;Br if not
004114  016305  000000G                 MOV     RT$TOP(R3),R5   ;Don't map beyond top of run-time
                                ;
                                ;  Return actual mapped size to user in R0
                                ;
004120  010537  000000G         2$:     MOV     R5,UR0          ;Return size of mapped region in user's R0
004124  160437  000000G                 SUB     R4,UR0
                                ;
                                ;  Finished
                                ;
004130  000207                          RETURN
```

```
                                        .SBTTL  ESPLIT -- Enable user D-space
                                ;-------------------------------------------------------------------
                                ;  Enable/Disable I and D space split
                                ;  When one or more of a job's PARs are mapped away to a shared run-time
                                ;  the root memory is not released, but is ordinarily inaccessible.  If
                                ;  I/D space split is enabled, then the root memory which was mapped
                                ;  away by a shared run-time can be accessed as D-space.  This is only valid
                                ;  on machines which support I and D space (have memory management register
                                ;  3 -- SR3FLG not 0 and have D-space APR's -- IDSFLG not 0).
                                ;
                                ;  Enabling D-space is done by setting the LSW11 bit which says the user
                                ;  wants to split, then calling SUTOP with the current size, which then
                                ;  calls SETMAP and copies the ordinary I-space APRs into the D-space APRs
                                ;  and enables user D-space.
                                ;
                                ;  The form of the EMT is:
                                ;       .BYTE   4,143
                                ;
004132  105737  000000G         ESPLIT: TSTB    IDSFLG          ;Does hardware support I and D space?
004136  001411                          BEQ     SPLITR          ;If not, br to split error return
004140  052761  000000G 000000G        BIS     #$UDSPC,LSW11(R1) ;Set flag to enable D-space mapping
004146  013700  000000G                 MOV     ODTBAS,R0       ;Point to word beyond normal address limit
004152  004737  000000G                 CALL    SUTOP           ;This will do the real work
004156  000137  000000G                 JMP     EMTXIT
                                ;
                                ;  Error, hardware does not support I and D space
                                ;
004162  012700  000006          SPLITR: MOV     #EC6,R0         ;Signal error code 1
004166  000137  000000G                 JMP     SETERR
                                ;
                                ;-------------------------------------------------------------------
                                ;  Disable D-space split, return all mapping to I-space
                                ;  This just requires turning off the hardware D-space enable bit,
                                ;  the D-space APRs will then be ignored so they don't require cleanup.
                                ;
                                ;  The form of the EMT is:
                                ;       .BYTE   5,143
                                ;
004172  105737  000000G         DSPLIT: TSTB    IDSFLG          ;Does hardware support I and D space?
004176  001771                          BEQ     SPLITR          ;Br if not
004200  042761  000000G 000000G        BIC     #$UDSPC,LSW11(R1) ;Clear user D-space in use flag
004206  042737  000000G 000000G        BIC     #USDSPC,@#SR3MMR ;And disable it in hardware
004214  000137  000000G                 JMP     EMTXIT
```

```
                                        .SBTTL  Define a PLAS region for fast mapping
                                ;-------------------------------------------------------------------
                                ;  Define a PLAS window so it can be quickly mapped using the TRAP
                                ;  instruction.  The region is not limited to 8Kb.  All checking related
                                ;  to PAR numbers and mapped length is assumed to have been done
                                ;  when creating the window.
                                ;
                                ;  EMT argument list:
                                ;       .BYTE   6,143
                                ;       .BYTE   W.NID,fmrnum    ;PLAS window ID, <0 to MAXSRD-1>
                                ;       .WORD   W.NRID          ;R.GID to which window is associated
                                ;
004220  113701  000002G         SSPLAS: MOVB    EMTBLK+2,R1     ;Get window ID
004224  005301                          DEC     R1              ;Make it 0 indexed
004226  020127  000000G                 CMP     R1,#NUMWCB      ;Is window ID in valid range? (1-NUMWCB)
004232  103403                          BLO     2$              ;Br if OK
004234  012700  000003          1$:     MOV     #EC3,R0         ;Invalid window ID
004240  000421                          BR      7$              ;Return with error
                                ;
                                ;  Verify that fast map region number is valid
                                ;
004242  113700  000003G         2$:     MOVB    EMTBLK+3,R0     ;Get fast map region number
004246  020027  000000G                 CMP     R0,#MAXSRD      ;Is number in valid range?
004252  103403                          BLO     3$              ;Br if OK
004254  012700  000002          21$:    MOV     #EC2,R0         ;Invalid fast map region #
004260  000411                          BR      7$              ;Return with error
                                ;
                                ;  Convert window ID into ptr to window control block
                                ;
004262  070127  000000G         3$:     MUL     #WC$$SZ,R1      ;And offset into window control blocks
004266  062701  000000G                 ADD     #WCBBAS,R1      ;Point to specified window control block
004272  016105  000000G                 MOV     WC$SIZ(R1),R5   ;Get # 64-byte blocks to be mapped
004276  001004                          BNE     4$              ;Br if window is allocated
004300  012700  000001          31$:    MOV     #EC1,R0         ;Else say window is undefined
004304  000137  000000G         7$:     JMP     SETERR          ;Return with error
                                ;
                                ;  Now move window control information into fast map cells
                                ;
004310  013702  000004G         4$:     MOV     EMTBLK+4,R2     ;Get ptr to region control block
004314  004737  004576'                 CALL    CHKRCB          ;Verify that it is valid
004320  103767                          BCS     31$             ;Br if invalid
004322  032762  000000G 000000G         BIT     #RC$LCG,RC$FLG(R2) ;Is this region swappable?
004330  001003                          BNE     41$             ;Br if not
004332  012700  000004                 MOV     #EC4,R0         ;Can't fast map swappable regions!
004336  000762                          BR      7$              ;Go do error
004340  005046                  41$:    CLR     -(SP)           ;Assume I-space mapping
004342  032762  000000G 000000G         BIT     #RC$DSP,RC$FLG(R2) ;Should we use D-space instead?
004350  001402                          BEQ     44$             ;Br if not
004352  112716  000000G                 MOVB    #SR.DSP,(SP)    ;If using D-space, init flag byte
                                ;
                                ;  Ensure that window won't exceed region bounds and update WCB cells
                                ;
004356  016204  000000G         44$:    MOV     RC$LEN(R2),R4   ;Get size of region
004362  166104  000000G                 SUB     WC$OFF(R1),R4   ;Figure maximum allowed window length
004366  020504                          CMP     R5,R4           ;Would request exceed region?
004370  103401                          BLO     46$             ;Br if not
004372  010405                          MOV     R4,R5           ;If so, just allow to end of region
```

Define a PLAS region for fast mapping

```
     58 004374  010561  000000G          46$:    MOV     R5,WC$LEN(R1)   ;And actual length into window control block
     59 004400  010537  000000G                  MOV     R5,URO          ;Return actual length to user
     60 004404  010504                          MOV     R5,R4           ;Copy window length
     61 004406  062704  000177                  ADD     #177,R4         ;Bound up to full par
     62 004412  072427  177771                  ASH     #-7,R4          ;Convert #64byte blocks to #8Kb pars
     63 004416  110461  000000G                 MOVB    R4,WC$NPR(R1)   ;And set into window control block
     64                                     ;
     65                                     ;  Loop to update as many PARs as required
     66                                     ;
     67 004422  016204  000000G                 MOV     RC$BAS(R2),R4   ;Get base of region
     68 004426  066104  000000G                 ADD     WC$OFF(R1),R4   ;Add offset to PLAS window
     69 004432  116102  000000G                 MOVB    WC$PAR(R1),R2   ;Get PAR index # (2*PAR #)
     70                                     ;  At this point, R0 has the trap #, NOT the trap index
     71 004436  110061  000000G                 MOVB    R0,WC$TRP(R1)   ;Set trap # into WCB (for inswap update)
     72 004442  020027  000000G          42$:    CMP     R0,#MAXSRD      ;Have we used up all trap #'s?
     73 004446  103402                          BLO     45$             ;Br if still OK
     74 004450  005726                          TST     (SP)+           ;Else clean flag word off stack
     75 004452  000700                          BR      21$             ;And go report error
     76 004454  110260  000000G          45$:    MOVB    R2,SR$PX(R0)    ;Set PAR index into fast map cell
     77 004460  111660  000000G                 MOVB    (SP),SR$FLG(R0) ;Init flag byte for this region
     78 004464  006300                          ASL     R0              ;Convert trap # to word index
     79 004466  010160  000000G                 MOV     R1,SR$WCB(R0)   ;Set ptr to wcb in fast map cell
     80 004472  010460  000000G                 MOV     R4,SR$PAR(R0)   ;Set PAR value into fast map cell
     81 004476  010503                          MOV     R5,R3           ;Get # blocks that still need to be mapped
     82 004500  020327  000200                  CMP     R3,#200         ;More than we can map with 1 par?
     83 004504  101402                          BLOS    43$             ;Br if not
     84 004506  012703  000200                  MOV     #200,R3         ;Map through this full par
     85 004512  160305                  43$:    SUB     R3,R5           ;Get # blocks left to be mapped
     86 004514  060304                          ADD     R3,R4           ;Bump PAR value up for next time
     87 004516  005303                          DEC     R3              ;Convert # blocks to # of last valid block
     88 004520  000303                          SWAB    R3              ;Move # blocks to high byte
     89 004522  052703  000006                  BIS     #6,R3           ;Enable read/write access
     90 004526  042703  100271                  BIC     #100271,R3      ;Clear unused PDR bits, ED is always up
     91 004532  010360  000000G                 MOV     R3,SR$PDR(R0)   ;Set PDR value into fast map cell
     92 004536  005705                          TST     R5              ;Need to map any more?
     93 004540  001410                          BEQ     5$              ;Br if not
     94 004542  006200                          ASR     R0              ;If so, convert back to trap #
     95 004544  152760  000000G 000000G        BISB    #SR.MOR,SR$FLG(R0) ;Say there are more regions to map
     96 004552  005200                          INC     R0              ;Bump up to next trap #
     97 004554  062702  000002                  ADD     #2,R2           ;Bump par index up to next register
     98 004560  000730                          BR      42$             ;Loop through as many pars as needed
     99                                     ;
    100                                     ;  All necessary fast trap cells are set up, enable fast mapping and exit
    101                                     ;
    102 004562  005726                  5$:     TST     (SP)+           ;Clean I- D-space flag off stack
    103 004564  112737  000001  000000G        MOVB    #1,DOTRMP       ;Enable fast mapping
    104 004572  000137  000000G                 JMP     EMTXIT
```

```
                                        .SBTTL  CHKRCB -- Check address of Region Control Block
                                ;-----------------------------------------------------------------------
                                ;  CHKRCB is called to verify that the address of a region control
                                ;  block is valid.  This routine is a similar to that in TSPLAS.
                                ;
                                ;  Inputs:
                                ;   R2 = Region control block address to be checked
                                ;
004576  032702  000001          CHKRCB: BIT     #1,R2           ;Make sure address is even
004602  001014                          BNE     1$              ;Br if odd
004604  020227  0000000G              CMP     R2,#RCBBAS      ;Make sure it is within proper region
004610  103403                          BLO     2$
004612  020227  0000000G              CMP     R2,#RCBEND
004616  103410                          BLO     3$
004620  020237  0000000G          2$:     CMP     R2,SHRRCB       ;See if this is a shared RCB
004624  103403                          BLO     1$
004626  020237  0000000G              CMP     R2,SHRRCN
004632  101402                          BLOS    3$
                                ;
                                ;  Address is invalid
                                ;
004634  000261                  1$:     SEC
004636  000401                          BR      9$
                                ;
                                ;  Address is ok
                                ;
004640  000241                  3$:     CLC
                                ;
004642  000207                  9$:     RETURN
```

```
                                        .SBTTL  Spooling control
                                ;-------------------------------------------------------------------
                                ;  This EMT is used to control the spooling operation.
                                ;
                                ;  Form of the EMT argument block:
                                ;       .BYTE   chan,151
                                ;       .WORD   sub-function
                                ;       .WORD   value
                                ;
                                ;  Where "chan" is an open channel whose file is to be affected,
                                ;  and "sub-function" indicates which function is to be performed.
                                ;
004644  013703  0000000         SPLEMT: MOV     CHNADR,R3       ;Get address of specified channel
004650  032763  0000000 0000000         BIT     #CS$OPN,C.CSW(R3);Is the channel open?
004656  001004                          BNE     1$              ;Br if yes
004660  012700  000003                  MOV     #EC3,R0         ;Channel not open
004664  000137  0000000                 JMP     SETERR
                                ;
                                ;  Ignore this EMT if the channel is not opened to a spooled device
                                ;
004670  032763  0000000 0000000 1$:     BIT     #CS$SPL,C.CSW(R3);Is channel opened to a spooled device?
004676  001412                          BEQ     9$              ;Ignore the EMT if not
                                ;
                                ;  Get sub-function code and branch off to processing routine
                                ;
004700  013700  0000020                 MOV     EMTBLK+2,R0     ;Get EMT sub-function code
004704  006300                          ASL     R0              ;Convert to word-table index
004706  020027  000006                  CMP     R0,#MXSFX       ;Is index ok?
004712  103402                          BLO     2$              ;Br if ok
004714  000137  0000000                 JMP     BADEMT          ;Invalid sub-function code
004720  000170  005020'       2$:     JMP     @SPLFNV(R0)     ;Enter processing routine
004724  000137  0000000       9$:     JMP     EMTXIT
                                ;
                                ;  Set Hold/Nohold flag for spool file
                                ;
004730  105737  0000000       SPSTHL: TSTB    NSPLDV          ;Are there any spooled devices?
004734  001405                          BEQ     9$              ;Br if not
004736  013702  0000040                 MOV     EMTBLK+4,R2     ;Get [no]hold flag
004742                                  OCALL   SPLSHF          ;Set hold flag for file
004750  000137  0000000       9$:     JMP     EMTXIT          ;Finished
                                ;
                                ;  Set Hold/Nohold flag for spool file
                                ;
004754  105737  0000000       SPSTFL: TSTB    NSPLDV          ;Are there any spooled devices?
004760  001404                          BEQ     9$              ;Br if not
004762  013702  0000040                 MOV     EMTBLK+4,R2     ;Get [no]flagpage flag
004766  000137  0000000                 JMP     SPLSFF          ;Set flagpage flag for device
004772  000137  0000000       9$:     JMP     EMTXIT          ;Finished
                                ;
                                ;  Set Hold/Nohold flag for spool file
                                ;
004776  105737  0000000       SPSTWD: TSTB    NSPLDV          ;Are there any spooled devices?
005002  001404                          BEQ     9$              ;Br if not
005004  013702  0000040                 MOV     EMTBLK+4,R2     ;Get wide/narrow flag
005010  000137  0000000                 JMP     SPLSWF          ;Set wide/narrow flag for device
005014  000137  0000000       9$:     JMP     EMTXIT          ;Finished
                                ;
```

```
                                ;  Vector of routine addresses for sub-function codes
                                ;
005020  004730'         SPLFNV: .WORD   SPSTHL          ; 0 -- Set [no]hold flag for file
005022  004754'                 .WORD   SPSTFL          ; 1 -- Turn on flag pages for device
005024  004776'                 .WORD   SPSTWD          ; 2 -- Set flag page wide/narrow for device
        000006          MXSFX   =       .-SPLFNV        ;# functions
```

```
                                        .SBTTL  CL line control EMT's
                                ;-----------------------------------------------------------------------
                                ;
                                ;  Assorted CL EMT's -- some of which duplicate CL spfun services, but
                                ;  are also implemented as EMT's to avoid the requirement to have a
                                ;  channel open to the CL unit.
                                ;
                                ;  The form of the argument block is:
                                ;
                                ;       .BYTE   n,155
                                ;       .WORD   CL-unit-number                  ;0 to (CLTOTL-1)
                                ;       .WORD   time-sharing-line-number        ;Only used for CLASN
                                ;
005026                          CLEMT:
                                ;
                                ;  Require TERMINAL privilege to manipulate CL units
                                ;
005026  032737  000000G 000000G         BIT     #P2$TRM,PRIVC2  ;Is this user privileged?
005034  001003                          BNE     12$             ;Br if yes
005036  012700  000001                  MOV     #EC1,R0         ;Error 1 if user not privileged
005042  000436                          BR      30$
                                ;
                                ;  Check to see if the specified CL unit number is valid
                                ;
005044  013702  0000020         12$:    MOV     EMTBLK+2,R2     ;Get specified CL unit number
005050  020227  000000G                 CMP     R2,#CLTOTL      ;Is this in the proper range?
005054  103403                          BLO     1$              ;Br if ok
005056  012700  000002                  MOV     #EC2,R0         ;Error 2
005062  000426                          BR      30$
005064  006302                  1$:     ASL     R2              ;Convert CL unit # to table index
                                ;
                                ;  Now jump off to the individual processing routines
                                ;
005066  113700  000000G                 MOVB    EMTBLK,R0       ;Get subfunction code
005072  020027  000003                  CMP     R0,#CLEMMX      ;Is it valid?
005076  103402                          BLO     2$              ;Br if OK
005100  005000                          CLR     R0              ;Else return ECO
005102  000416                          BR      30$
                                ;
005104  006300                  2$:     ASL     R0              ;Convert subfunction code to index
005106  001416                          BEQ     3$              ;Br if we don't need to check for hdw or priv
005110  032737  000000G 000000G         BIT     #P0$BYP,PRIVC0  ;Does issuer have BYPASS privilege?
005116  001003                          BNE     21$             ;Br if yes
005120  012700  000010                  MOV     #EC8,R0         ;Oh, too bad. Must have BYPASS
005124  000405                          BR      30$
005126  016201  000000G         21$:    MOV     CL$LIX(R2),R1   ;Get index of T-S line CL is connected to
005132  001004                          BNE     3$              ;Br if it is connected to a line
005134  012700  000007                  MOV     #EC7,R0         ;Oops, not connected to hardware!
005140  000137  000000G         30$:    JMP     SETERR
                                ;
005144  000170  005150'          3$:     JMP     @CLCODE(R0)     ;Enter the processing routine
                                ;
005150  005156'                 CLCODE: .WORD   CLASN           ;0  Assign a CL unit to a line
005152  005674'                         .WORD   CLCLR           ;1  Clear XOFF status (=== SPFUN 201)
005154  005710'                         .WORD   CLRES           ;2  Reset CL unit     (=== SPFUN 265)
        000003                  CLEMMX  = <.-CLCODE>/2
```

```
                        .SBTTL          CLASN -- Assign a CL unit to a line
                ;-----------------------------------------------------------------------
                ;  Assign a CL unit to a line.
                ;  The form of the EMT is:
                ;
                ;       .BYTE   0,155
                ;       .WORD   CL_unit_number
                ;       .WORD   line_number
                ;
005156          CLASN:
                ;
                ;  Check to see if the specified line number is ok
                ;
005156  013704  000004G         MOV     EMTBLK+4,R4     ;Get specified line number
005162  001451                  BEQ     5$              ;LINE=0 ==> Free this unit
005164  006304                  ASL     R4              ;Convert line # to line table index
005166  020427  000000G         CMP     R4,#LSTHL       ;Must be a valid hardware line
005172  101042                  BHI     2$              ;Br if invalid
005174  032764  000000G 000000G BIT     #$HARD,LSW3(R4) ;Is this line connected to hardware?
005202  001436                  BEQ     2$              ;Br if not
005204  020264  000000G         CMP     R2,LCLUNT(R4)   ;Is this CL unit already assigned to this line
005210  001535                  BEQ     22$             ;Br if yes -- nothing else needs to be done
005212  032764  000000G 000000G BIT     #$DEAD,LSW3(R4) ;Is this line installed?
005220  001027                  BNE     2$              ;Br if not
005222  005764  000000G         TST     LCLUNT(R4)      ;Is this line in use by another CL unit?
005226  002404                  BLT     4$              ;Br if not
005230  012700  000004          MOV     #EC4,R0         ;Error 4 if line already being used by CL
005234  000137  000000G 30$:    JMP     SETERR
005240  020427  000000G 4$:     CMP     R4,#LSTPL       ;Connecting to a primary time-sharing line?
005244  101007                  BHI     3$              ;Br if not
005246  032764  000000G 000000G BIT     #$DILUP,LSW(R4) ;Is this line in use by time-sharing user?
005254  001414                  BEQ     5$              ;Br if not
005256  012700  000005          MOV     #EC5,R0         ;Error 5 if line is active as a time-sharing
005262  000764                  BR      30$
005264  020427  000000G 3$:     CMP     R4,#FSTIOL      ;Are we assigning to a reserved CL line?
005270  103403                  BLO     2$              ;Br if not
005272  020427  000000G         CMP     R4,#LSTIOL      ;Is this a valid CL line?
005276  101403                  BLOS    5$              ;Br if yes
005300  012700  000003 2$:      MOV     #EC3,R0         ;Error 3 if invalid line number
005304  000753                  BR      30$
                ;
                ;  The line number is ok.
                ;  Deassign the CL unit from its current line association.
                ;
005306  016203  000000G 5$:     MOV     CL$LIX(R2),R3   ;Get line # currently assoc with this CL unit
005312  001471                  BEQ     6$              ;Br if line is not currently assigned
005314  005762  000000G         TST     CL$XLN(R2)      ;Is CL unit in use for cross connect?
005320  001005                  BNE     23$             ;Br if yes
005322  016200  000000G         MOV     CL$RQH(R2),R0   ;Anything in read queue head
005326  056200  000000G         BIS     CL$WQH(R2),R0   ;or write queue head for this CL unit?
005332  001403                  BEQ     11$             ;Br if not
005334  012700  000006 23$:     MOV     #EC6,R0         ;Error 6 if CL unit is currently busy
005340  000735                  BR      30$
005342  004737  006026' 11$:     CALL    CLWAIT          ;Wait for output characters to be sent
005346  010205                  MOV     R2,R5           ;Get CL unit index to R5
005350                          OCALL   CLREST          ;Reset CL unit status
005356  042762  000000G 000000G BIC     #CO$DTR,CL$OPT(R2) ;Request DTR down
```

```
     58 005364                                          OCALL   SETDTR          ;Call CL routine to drop DTR
     59 005372  012763  177777  000000G                MOV     #-1,LCLUNT(R3)  ;Say this line not assoc with a CL unit
     60 005400  016300  000000G                         MOV     LCXTBL(R3),R0   ;Get pointer to translation table for line
     61 005404  001401                                  BEQ     24$             ;Br if no translation table
     62 005406  005010                                  CLR     (R0)            ;Say no translation in effect
     63 005410  020327  000000G              24$:       CMP     R3,#LSTPL       ;Is this a time-sharing line
     64 005414  101022                                  BHI     7$              ;Br if not
     65 005416  012763  000000G  000000G               MOV     #NEDCHR,LOUTIR(R3);Set T/S terminal output interrupt rtn
     66 005424  012763  000000G  000000G               MOV     #TTINCP,LINIR(R3);Set T/S terminal input interrupt rtn
     67 005432  032763  000000G  000000G               BIT     #$AUTO,ILSW2(R3);Is this line suppose to autobaud?
     68 005440  001416                                  BEQ     6$              ;Br if not
     69 005442  010301                                  MOV     R3,R1           ;Get number of line being released
     70 005444  012700  000000G                         MOV     #S9600,R0       ;Get 9600 baud speed code
     71 005450  004737  000000G                         CALL    SETSPD          ;Set line speed
     72 005454  113701  000000G                         MOVB    CORUSR,R1       ;Get back our line index number
     73 005460  000406                                  BR      6$
     74 005462  012763  000000G  000000G  7$:           MOV     #LINRTS,LOUTIR(R3);Set dummy routine to do RTS
     75 005470  012763  000000G  000000G               MOV     #LINRTS,LINIR(R3)
     76                                     ;
     77                                     ;  Associate the specified line with the CL unit
     78                                     ;
     79 005476  010462  000000G              6$:        MOV     R4,CL$LIX(R2)   ;Set # of line associated with CL unit
     80 005502  001002                                  BNE     21$             ;Br if not freeing this CL unit
     81 005504  000137  000000G              22$:       JMP     EMTXIT          ;LINE=0 ==> Free this CL unit
     82 005510  010264  000000G              21$:       MOV     R2,LCLUNT(R4)   ;Set CL unit in line table
     83 005514  012764  000000G  000000G               MOV     #CLOTIR,LOUTIR(R4);Set output interrupt service routine
     84 005522  012764  000000G  000000G               MOV     #CLINCP,LINIR(R4);Set terminal input service routine
     85 005530  052764  000000G  000000G               BIS     #$SXON,LSW10(R4);Send XON when we start
     86                                     ;
     87                                     ;  Set some option flags for this CL unit
     88                                     ;
     89 005536  016403  000000G                         MOV     LCXTBL(R4),R3   ;Get pointer to translation table for line
     90 005542  001401                                  BEQ     25$             ;Br if no translation table
     91 005544  005013                                  CLR     (R3)            ;Say no translation in effect
     92 005546  016403  000000G              25$:       MOV     ILSW2(R4),R3    ;Get option flags for the line
     93 005552  012700  000000G                         MOV     #CO$DEF,R0      ;Get default CL option flags
     94 005556  032703  000000G                         BIT     #$TAB,R3        ;Does this device support tabs?
     95 005562  001402                                  BEQ     8$              ;Br if not
     96 005564  052700  000000G                         BIS     #CO$TAB,R0      ;Set CL tab flag
     97 005570  032703  000000G              8$:        BIT     #$FORM,R3       ;Does this device suport form-feeds?
     98 005574  001402                                  BEQ     9$              ;Br if not
     99 005576  052700  000000G                         BIS     #CO$FF,R0       ;Set CL form feed flag
    100 005602  032703  000000G              9$:        BIT     #$8BIT,R3       ;Does this device want 8-bit support?
    101 005606  001402                                  BEQ     10$             ;Br if not
    102 005610  052700  000000G                         BIS     #CO$8BT,R0      ;Enable 8 bit support
    103 005614  010062  000000G              10$:       MOV     R0,CL$OPT(R2)   ;Initialize CL options for this device
    104                                     ;
    105                                     ;  Initialize status for this CL unit
    106                                     ;
    107 005620  005062  000000G                         CLR     CL$STA(R2)      ;Init status flags
    108 005624  005062  000000G                         CLR     CL$COL(R2)      ;Current column number
    109 005630  012762  000102  000000G                MOV     #66.,CL$LEN(R2) ;# lines per page
    110 005636  005062  000000G                         CLR     CL$LIN(R2)      ;Current line number
    111 005642  005062  000000G                         CLR     CL$SKP(R2)      ;Lines to skip at bottom of page
    112 005646  005062  000000G                         CLR     CL$WID(R2)      ;Line width
    113 005652  005062  000000G                         CLR     CL$EPN(R2)      ;No end-of-file form-feeds
    114 005656  016200  000000G                         MOV     CL$EPS(R2),R0   ;Get pointer to ENDSTRING buffer
```

```
    115 005662  105010                          CLRB    (R0)            ;No ENDSTRING
    116 005664  005062  0000000                  CLR     CL$EPP(R2)
    117                             ;
    118                             ;  Finished
    119                             ;
    120 005670  000137  0000000     20$:         JMP     EMTXIT          ;Finished assigning the CL unit
```

```
                                        .SBTTL          CLCLR -- Clear XOFF sent and send XON
                                ;-----------------------------------------------------------------
                                ;  This routine is the EMT equivalent of CL SPFUN 201
                                ;
                                ;  Emt argument block:
                                ;       .BYTE   1,155
                                ;       .WORD   CL-unit-number
                                ;
                                ;  Inputs:
                                ;       R1      TS line in use as CL unit
                                ;       R2      CL unit index
                                ;
005674                          CLCLR:
005674  010205                          MOV     R2,R5           ;Need to get CL unit index into R5
005676                                  OCALL   CLCLER          ;Call TSCLO routine to clear CL unit
005704  000137  0000006                 JMP     EMTXIT
                                ;
                                        .SBTTL          CLRES -- Reset CL line
                                ;-----------------------------------------------------------------
                                ;  This routine is the EMT equivalent of CL SPFUN 265
                                ;
                                ;  EMT argument block:
                                ;       .BYTE   2,155
                                ;       .WORD   CL-unit-number
                                ;
                                ;  Inputs:
                                ;       R1      TS line in use as CL unit
                                ;       R2      CL unit index
005710                          CLRES:
005710  010205                          MOV     R2,R5           ;Need to get CL unit index into R5
005712                                  OCALL   CLREST          ;Call TSCLO routine to reset CL unit
005720  000137  0000006                 JMP     EMTXIT
```

```
                                        .SBTTL  CLCLOS -- Close channel opened to CL unit
                                ;-------------------------------------------------------------------
                                ;  This routine is called to determine if the channel being closed is
                                ;  assigned to a CL unit.  If it is, the job is suspended until all output
                                ;  to the CL unit has been completed.
                                ;
005724  010246                  CLCLOS: MOV     R2,-(SP)
005726  010546                          MOV     R5,-(SP)
                                ;
                                ;  Determine if this channel is opened to a CL unit
                                ;
005730  013702  000000G                 MOV     CHNADR,R2       ;Get address of current channel block
005734  016200  000000G                 MOV     C.CSW(R2),R0    ;Get channel status word
005740  032700  000000G                 BIT     #CS$SPL,R0      ;Is this a spooled device?
005744  001025                          BNE     9$              ;Br if yes -- Don't wait for output to finish
005746  116202  000000G                 MOVB    C.DEVQ(R2),R2   ;Get unit number
005752  042702  177770                  BIC     #^C<7>,R2       ;Clear all but unit number
005756  042700  000000C                 BIC     #^C<CS$NMX>,R0  ;Mask out all but device table index
005762  020037  000000G                 CMP     R0,CLDEVX       ;Is this a CL device?
005766  001405                          BEQ     1$              ;Br if yes
005770  020037  000000G                 CMP     R0,C1DEVX       ;Is this a C1 device?
005774  001011                          BNE     9$              ;Br if not
005776  062702  000010                  ADD     #8.,R2          ;Bias unit number by 8 for C1
                                ;
                                ;  This is a CL device being closed.
                                ;  Wait for all output to be completed.
                                ;
006002  006302                  1$:     ASL     R2              ;Convert to word table index
006004  004737  006026'                 CALL    CLWAIT          ;Wait for all output to finish
006010  010205                          MOV     R2,R5           ;Get CL unit index to R5
006012                                  OCALL   CLREST          ;Reset CL unit status
                                ;
                                ;  Finished
                                ;
006020  012605                  9$:     MOV     (SP)+,R5
006022  012602                          MOV     (SP)+,R2
006024  000207                          RETURN
```

```
 1                                              .SBTTL  CLWAIT -- Wait for CL output to finish
 2                                      ;-----------------------------------------------------------------------
 3                                      ;  This routine is called to check to see if all characters which have
 4                                      ;  been queued for transmission on a CL line have actually been transmitted.
 5                                      ;  If there are still pending characters the job is suspended for a while
 6                                      ;  to give the characters time to be sent.
 7                                      ;
 8                                      ;  Inputs:
 9                                      ;   R2 = CL unit index
10                                      ;
11 006026  010346                      CLWAIT: MOV     R3,-(SP)
12                                      ;
13                                      ;  Set maximum number of times we will wait for output to finish
14                                      ;
15 006030  012703  000024                       MOV     #20.,R3         ;Set max iteration count
16                                      ;
17                                      ;  See if there are any output characters pending for this CL unit
18                                      ;
19 006034  026262  000000G 000000G      1$:     CMP     CL$ORS(R2),CL$ORA(R2) ;Are there any pending output chars?
20 006042  001013                               BNE     2$              ;Br if yes
21 006044  016200  000000G                      MOV     CL$LIX(R2),R0   ;Get index # of line CL is connected to
22 006050  001421                               BEQ     9$              ;Br if CL unit not connected to line
23 006052  032762  000000G 000000G             BIT     #CM$EFP,CL$STA(R2);Are we doing end-of-file processing?
24 006060  001004                               BNE     2$              ;Br if yes
25 006062  032760  000000G 000000G             BIT     #$XCHAR,LSW3(R0);Is output transmission to that line active?
26 006070  001411                               BEQ     9$              ;Br if not
27                                      ;
28                                      ;  There are some pending output characters.
29                                      ;  Suspend job for a little while and then check again.
30                                      ;
31 006072  012746  000000G             2$:     MOV     #CLWTIM,-(SP)   ;Push pointer to time value cells
32 006076  012746  012000                       MOV     #<24*400>,-(SP) ;Push EMT argument code
33 006102  010600                               MOV     SP,R0           ;Point to EMT arg block we built on stack
34 006104  104375                               EMT     375             ;Do .TWAIT
35 006106  062706  000004                       ADD     #4.,SP          ;Pop EMT argument block from stack
36                                      ;
37                                      ;  Go back and check for output pending
38                                      ;
39 006112  077330                               SOB     R3,1$           ;Go back and check for output pending
40                                      ;
41                                      ;  There is no output pending or we have wait as long as we are willing
42                                      ;
43 006114  012603                      9$:     MOV     (SP)+,R3
44 006116  000207                               RETURN
```

```
                                        .SBTTL  Performance monitor emt's
                                ;-------------------------------------------------------------------
                                ;  The following EMT's are used to control the TSX-Plus performance
                                ;  monitor system.
                                ;
                                ;  The EMT sub-function stored in byte 0 of the argument block controls
                                ;  the action of the emt.  The valid sub-functions are:
                                ;
                                ;  0 = Initialize the performance monitor system.
                                ;  1 = Start doing performance monitoring.
                                ;  2 = Suspend performance monitoring.
                                ;  3 = Stop performance monitor and return result values.
                                ;
006120  113705  000000G       PMEMT:  MOVB    EMTBLK,R5       ;GET SUB-FUNCTION CODE FOR EMT
006124  120527  000003                CMPB    R5,#3           ;VALID SUB-FUNCTION CODE?
006130  101402                       BLOS    1$              ;BR IF OK
006132  000137  000000G              JMP     BADEMT          ;INVALID EMT CODE
006136  006305                1$:     ASL     R5              ;CONVERT TO WORD-TABLE INDEX
006140  000175  006144'              JMP     @PMRVEC(R5)     ;ENTER PROCESSING ROUTINE
                                ;
006144  006154'               PMRVEC: .WORD   PMINIT          ;0 -- INITIALIZE
006146  006366'                       .WORD   PMSTRT          ;1 -- START MONITORING
006150  006402'                       .WORD   PMSTOP          ;2 -- SUSPEND MONITORING
006152  006416'                       .WORD   PMEND           ;3 -- END ANALYSIS
```

```
                                ;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                                ;  Initialize performance monitoring.
                                ;
                                ;  Arguments:
                                ;    1. Base address of region being monitored.
                                ;    2. Top address of region being monitored.
                                ;    3. Number of bytes per cell in histogram.
                                ;    4. PF$ control flags.
                                ;
                                ;  Errors:
                                ;    0 = Performance monitor already in use.
                                ;    1 = Performance monitor not generated in.
                                ;
006154  005737  000000G       PMINIT: TST     PMCELS          ;WAS PERFORMANCE MONITOR GENNED IN?
006160  001004                        BNE     5$              ;BR IF YES
006162  012700  000001                MOV     #EC1,R0         ;RETURN ERROR CODE 1
006166  000137  000000G               JMP     SETERR
006172  113702  000000G       5$:     MOVB    PMUSER,R2       ;IS ANYONE DOING PERFORMANCE MONITORING NOW?
006176  001411                        BEQ     1$              ;BR IF NOT
006200  120237  000000G               CMPB    R2,CORUSR       ;ARE WE THE ONE?
006204  001406                        BEQ     1$              ;BR IF YES
006206  005000                        CLR     R0              ;PM IN USE BY SOMEONE ELSE -- RETURN ERROR 0
006210  006202                        ASR     R2              ;RETURN # OF JOB THAT OWNS PM IN USER'S R0
006212  010237  000000G               MOV     R2,UR0
006216  000137  000000G               JMP     SETERR
006222  113737  000000G 000000G 1$:   MOVB    CORUSR,PMUSER   ;SAY WE ARE USING PM FACILITY
006230  005037  000000G               CLR     PMRUN           ;DON'T START MONITORING YET
                                ;
                                ;  Save information about region being monitored.
                                ;
006234  013705  000002G               MOV     EMTBLK+2,R5     ;GET BASE ADDRESS OF REGION
006240  010537  000000G               MOV     R5,PMBASE
006244  013703  000004G               MOV     EMTBLK+4,R3     ;GET TOP ADDRESS OF REGION
006250  010337  000000G               MOV     R3,PMTOP
006254  013737  000010G 000000G       MOV     EMTBLK+10,PMFLGS;GET CONTROL FLAGS
006262  160503                        SUB     R5,R3           ;GET # BYTES IN REGION BEING MONITORED
006264  000241                        CLC
006266  006003                        ROR     R3              ;CONVERT TO # WORDS
006270  005002                        CLR     R2              ;SET FOR DIVIDE
006272  071237  000000G               DIV     PMCELS,R2       ;DIVIDE BY # CELLS IN PM DATA AREA
006276  005703                        TST     R3              ;IS REMAINDER ZERO?
006300  001401                        BEQ     2$              ;BR IF YES
006302  005202                        INC     R2              ;ROUND UP QUOTIENT IF NOT
006304  006302                2$:     ASL     R2              ;GET # BYTES PER PM CELL
006306  020237  000006G               CMP     R2,EMTBLK+6     ;COMPARE WITH REQUESTED # BYTES PER CELL
006312  103002                        BHIS    4$              ;USE LARGER VALUE
006314  013702  000006G               MOV     EMTBLK+6,R2
006320  010237  000000G       4$:     MOV     R2,PMNBPC       ;NUMBER OF BYTES PER CELL
                                ;
                                ;  Zero performance monitoring count cells.
                                ;  (Enter system state so that we are running on interrupt stack and can
                                ;  access PM data area through PAR6.)
                                ;
006324  012700  006362'               MOV     #6$,R0          ;GO TO 6$ ON EXIT FROM SYSTEM STATE
006330  004737  000000G               CALL    ENSYS           ;ENTER SYSTEM STATE
006334  000000                        .WORD   FP$MOV          ;Set fork processing priority
006336  013700  000000G               MOV     PMCELS,R0       ;GET # CELLS
```

Performance monitor emt's

```
     58 006342  012705  000000G                    MOV     #VPAR6,R5          ;ACCESS AREA THROUGH PAR6 REGION
     59 006346  013737  000000G 000000G             MOV     PMPAR,@#KPAR6      ;MAP TO PERFORMANCE MONITOR DATA AREA
     60 006354  005025                       3$:    CLR     (R5)+              ;ZERO DATA VECTOR
     61 006356  077002                              SOB     R0,3$
     62 006360  000207                              RETURN                     ;EXIT SYSTEM STATE -- GO TO 6$
     63                                     ;
     64                                     ;  Finished
     65                                     ;
     66 006362  000137  000000G             6$:     JMP     EMTXIT             ;EXIT FROM EMT
```

```
                                ;-------------------------------------------------------------------------------
                                ;  Start monitoring execution.
                                ;
006366  004737  006616'        PMSTRT: CALL    PMCHKU          ;MAKE SURE WE ARE PM USER
006372  005237  0000006G              INC     PMRUN           ;START MONITORING
006376  000137  0000006G              JMP     EMTXIT
                                ;-------------------------------------------------------------------------------
                                ;  Suspend performance monitoring.
                                ;
006402  004737  006616'        PMSTOP: CALL    PMCHKU          ;MAKE SURE WE ARE PM USER
006406  005037  0000006G              CLR     PMRUN           ;SUSPEND PM MONITORING
006412  000137  0000006G              JMP     EMTXIT
                                ;-------------------------------------------------------------------------------
                                ;  Return results of performance monitoring.
                                ;
                                ;  Arguments:
                                ;   1. Address of vector to receive parameter values.
                                ;   2. Address of area to receive count vector.
                                ;   3. Size of area to receive count vector (# bytes).
                                ;
                                ;  Values returned in parameter vector:
                                ;   1. Base address of monitored region.
                                ;   2. Top address of monitored region.
                                ;   3. Number of bytes per count cell.
                                ;   4. PF$ control and status flags.
                                ;
                                ;  Errors:
                                ;   0 = This job is not doing a performance monitor.
                                ;   1 = Area for count vector is too small.
                                ;
006416  004737  006616'        PMEND:  CALL    PMCHKU          ;MAKE SURE WE ARE PM USER
006422  005037  0000006G              CLR     PMRUN           ;STOP MONITORING
006426  005037  0000006G              CLR     PMUSER          ;SAY PERFORMANCE MONITOR IS FREE
                                ;
                                ;  Return parameter values
                                ;
006432  013701  0000020               MOV     EMTBLK+2,R1     ;GET ADDRESS OF USER'S RECEIVING AREA
006436  010100                        MOV     R1,R0           ;VALIDATE THE ADDRESS
006440  004737  0000006G              CALL    VALADW
006444  013746  0000006G              MOV     PMBASE,-(SP)    ;RETURN BASE ADDRESS
006450  106621                        MTPD    (R1)+
006452  013746  0000006G              MOV     PMTOP,-(SP)     ;RETURN TOP ADDRESS
006456  106621                        MTPD    (R1)+
006460  013746  0000006G              MOV     PMNBPC,-(SP)    ;NUMBER OF BYTES PER CELL
006464  106621                        MTPD    (R1)+
006466  013746  0000006G              MOV     PMFLGS,-(SP)    ;STATUS FLAGS
006472  106621                        MTPD    (R1)+
                                ;
                                ;  Return count values.
                                ;
                                ;  Enter system state so that we are running on the interrupt stack and
                                ;  can map to PM data area through PAR6.
                                ;
006474  012700  006576'               MOV     #3$,R0          ;GO TO 3$ ON RETURN FROM SYSTEM STATE
006500  004737  0000006G              CALL    ENSYS           ;ENTER SYSTEM STATE
```

Performance monitor emt's

```
                                        .WORD   FP$MOV          ;Fork processing priority
006506  013702  000004G                 MOV     EMTBLK+4,R2     ;GET ADDRESS OF USER'S BUFFER
006512  010200                          MOV     R2,R0           ;VALIDATE THE ADDRESS
006514  004737  000000G                 CALL    VALADW
006520  013703  000006G                 MOV     EMTBLK+6,R3     ;GET # BYTES IN USER'S AREA
006524  006203                          ASR     R3              ;CONVERT TO # WORDS
006526  013704  000000G                 MOV     PMBASE,R4       ;GET ADDRESS OF BASE OF MONITORED REGION
006532  012705  000000G                 MOV     #VPAR6,R5       ;ACCESS PM DATA AREA THROUGH PAR6 REGION
006536  013737  000000G 000000G         MOV     PMPAR,@#KPAR6   ;SET PAR6 TO MAP TO PM DATA AREA
006544  005037  006614'                 CLR     PMOVFR          ;SAY USER BUFFER HAS NOT OVERFLOWED
006550  012546                  1$:     MOV     (R5)+,-(SP)     ;GET COUNT VALUE
006552  106622                          MTPD    (R2)+           ;MOVE TO USER'S AREA
006554  063704  000000G                 ADD     PMNBPC,R4       ;ADVANCE ADDRESS COUNTER
006560  020437  000000G                 CMP     R4,PMTOP        ;RETURNED ALL VALUES?
006564  103003                          BHIS    2$              ;BR IF YES
006566  077310                          SOB     R3,1$           ;CONTINUE MOVING
006570  005237  006614'                 INC     PMOVFR          ;SIGNAL OVERFLOW OF USER'S AREA
006574  000207                  2$:     RETURN                  ;EXIT SYSTEM STATE -- GO TO 3$
                                ;
                                ;  Finished
                                ;
006576  013700  006614'          3$:     MOV     PMOVFR,R0       ;GET USER OVERFLOW ERROR FLAG
006602  001402                          BEQ     4$              ;BRANCH IF NO OVERFLOW
006604  000137  000000G                 JMP     SETERR          ;ELSE REPORT ERROR TO USER
006610  000137  000000G          4$:     JMP     EMTXIT          ;FINISHED
006614  000000                  PMOVFR: .WORD   0               ;USER BUFFER OVERFLOW ERROR FLAG
                                ;-------------------------------------------------------------------------------
                                ;  Subroutine to check if the current user is doing a performance monitor.
                                ;  If not an error return with code 0 is taken.
                                ;
006616  123737  000000G 000000G PMCHKU: CMPB    CORUSR,PMUSER   ;ARE WE PM USER?
006624  001410                          BEQ     1$              ;BR IF YES
006626  005000                          CLR     R0              ;RETURN ERROR CODE 0
006630  113702  000000G                 MOVB    PMUSER,R2       ;RETURN # OF USER WHO OWNS PM IN R0
006634  006202                          ASR     R2
006636  010237  000000G                 MOV     R2,UR0
006642  000137  000000G                 JMP     SETERR
006646  000207                  1$:     RETURN
```

```
                                        .SBTTL  CKACJB -- See if access to another job is allowed
                                ;-------------------------------------------------------------------------
                                ;  Determine if the current job is privileged to access another job.
                                ;
                                ;  Inputs:
                                ;   R2 = Job index number of job we wish to access.
                                ;
                                ;  Outputs:
                                ;   C-flag cleared ==> Access ok
                                ;   C-flag set ==> Access disallowed
                                ;
006650  010146                  CKACJB: MOV     R1,-(SP)
                                ;
                                ;  Always allow access to our own job
                                ;
006652  120237  000000G                 CMPB    R2,CORUSR       ;Affecting our own job?
006656  001437                          BEQ     7$              ;Br if yes
                                ;
                                ;  If we have WORLD privilege we can access any job
                                ;
006660  032737  000000G 000000G        BIT     #P2$WRL,PRIVC2  ;Do we have WORLD privilege?
006666  001033                          BNE     7$              ;Br if yes
                                ;
                                ;  Always allow access to our virtual lines and children jobs.
                                ;
006670  113701  000000G                 MOVB    CORUSR,R1       ;Get our job index number
006674  126162  000000G 000000G        CMPB    LNPRIM(R1),LNPRIM(R2) ;Do we have the same primary line?
006702  001425                          BEQ     7$              ;Br if yes
006704  026201  000000G                 CMP     LPARNT(R2),R1   ;Are we parent of this job?
006710  001422                          BEQ     7$              ;Br if yes
                                ;
                                ;  See if project numbers of jobs match
                                ;
006712  026162  000000G 000000G        CMP     LPROJ(R1),LPROJ(R2) ;Do project numbers match?
006720  001014                          BNE     6$              ;Br if not -- We cannot change job
006722  032737  000000G 000000G        BIT     #P2$GRP,PRIVC2  ;Do we have GROUP privilege?
006730  001012                          BNE     7$              ;Br if yes
                                ;
                                ;  Project numbers match, check programmer numbers.
                                ;
006732  026162  000000G 000000G        CMP     LPROG(R1),LPROG(R2) ;Do programmer numbers match?
006740  001004                          BNE     6$              ;Br if not
006742  032737  000000G 000000G        BIT     #P2$SAM,PRIVC2  ;Do we have SAME privilege?
006750  001002                          BNE     7$              ;Br if yes
                                ;
                                ;  We cannot access the job
                                ;
006752  000261                  6$:     SEC                     ;Signal failure on return
006754  000401                          BR      9$
                                ;
                                ;  We can access the job
                                ;
006756  000241                  7$:     CLC                     ;Signal success on return
                                ;
                                ;  Finished
                                ;
006760  012601                  9$:     MOV     (SP)+,R1
```

CKACJB -- See if access to another job is allowed

```
     58 006762  000207                          RETURN
     59         000001                          .END
Errors detected:  0

*** Assembler statistics


Work  file  reads:  0
Work  file writes:  0
Size of work file:  9639 Words  ( 38 Pages)
Size of core pool:  18176 Words  ( 71 Pages)
Operating  system:  RT-11

Elapsed time:  00:00:43.09
,LP:TSEM4=DK:TSEM4/C/N:SYM
```

Cross reference table (CREF V05.05)

```
$8BIT     1-78      28-100
$AUTO     1-77      12-7       28-67
$CTRLO    1-69      13-42
$CTRLS    1-69      13-41
$DEAD     1-75      28-23
$DEBUG    1-64       3-210
$DETCH    1-67      11-9       16-19
$DILUP    1-67      10-6       12-5       13-25      17-34      17-70      28-31
$DISCN    1-71      13-44
$FORM     1-62       3-146     28-97
$HARD     1-75      28-19
$INKMN    1-56       2-7        3-8
$IOMAP    1-55       1-63       3-183
$LOFCF    1-53      16-21
$NOIN     1-53      16-23
$NOVLN    1-53      16-17
$RBRK     1-54      12-13
$RDSAV    1-57       3-23       3-177
$RNIOP    1-63       3-181
$RNMLK    1-63       3-191      3-193
$SCOPE    1-61       3-140
$SETRN    1-63       3-159
$SUCF     1-53      16-21
$SUSPN    1-67      10-13      10-18      13-43
$SXON     1-78      28-85
$TAB      1-61       1-78       3-143     28-94
$UDSPC    1-84      23-21      23-41
$VBELL    1-52      16-129     16-161
$VIRJB    1-61       3-131
$VNOTT    1-52      16-132
$XCHAR    1-81      31-25
...V1     3-66       3-66       3-74       3-74       3-221      3-221      6-30
...V2     3-66       3-66       3-66#      3-66#      3-74       3-74#      3-221      3-221#
ABORT     1-64       3-224
BADEMT    1-56       2-9        18-19      26-30      32-17
C.CSW     1-74      26-14      26-21      30-13
C.DEVQ    1-80      30-16
C1DEVX    1-80      30-21
CHAIN     1-60       3-83       3-87       3-88       3-107
CHKRCB   24-40      25-9#
CHNADR    1-74      26-13      30-12
CHNSIZ    1-59       3-51       3-53       3-55
CINDAT    1-60       3-105*     3-106*     3-109      5-14
CINFLG    1-60       3-84       3-86*      3-220*
CINMOV    3-91       3-112      5-8#
CKACJB   13-5       17-88      35-12#
CL$COL    1-79      28-108*
CL$EPN    1-80      28-113*
CL$EPP    1-80      28-116*
CL$EPS    1-80      28-114
CL$LEN    1-79      28-109*
CL$LIN    1-79      28-110*
CL$LIX    1-76      27-46      28-45      28-79*     31-21
CL$OPT    1-79      11-38*     28-57*     28-103*
CL$ORA    1-81      31-19
CL$ORS    1-81      31-19
```

Cross reference table (CREF V05.05)

```
CL$RQH    1-76      28-49
CL$SKP    1-79      28-111*
CL$STA    1-68       1-79      11-28*    11-29*    28-107*   31-23
CL$WID    1-79      28-112*
CL$WQH    1-76      28-50
CL$XLN    1-68      11-27*    28-47
CLASN    27-53      28-10#
CLCLER    1-58      29-15
CLCLOS    1-44      30-7#
CLCLR    27-54      29-13#
CLCODE   27-51      27-53#    27-56
CLDEVX    1-80      30-19
CLEMMX   27-35      27-56#
CLEMT     1-44      27-14#
CLINCP    1-77      28-84
CLOTIR    1-77      28-83
CLRES    27-55      29-29#
CLREST    1-57      28-56     29-31     30-31
CLTOTL    1-75      27-26
CLWAIT   28-54      30-29     31-11#
CLWTIM    1-54      31-31
CLXBRK    1-69      13-35
CLXCON    2-34      11-5#
CLXICP    1-68      11-34
CM$BRK    1-68      11-28
CM$EFP    1-81      31-23
CM$ON     1-68      11-29
CO$8BT    1-78      28-102
CO$DEF    1-78      28-93
CO$DTR    1-54      11-38     28-57
CO$FF     1-78      28-99
CO$TAB    1-78      28-96
CORUSR    1-72      17-55     28-72     33-20     33-26     34-89     35-16     35-26
CPLEMT    1-60       3-123*
CS$NMX    1-80      30-18
CS$OPN    1-74      26-14
CS$SPL    1-80      26-21     30-14
CSHALC    1-54      14-4
CSHINI    1-54      14-6
CSHNEW    2-29      14-4#
CSIARE    1-59       3-66      3-194     6-30
CUPARO    1-74      20-24*
CUPDRO    1-74      20-38*
CURRDB    1-55      19-17*    19-53*    22-25
CXTRMN    1-58       1-62      3-49      3-152
DBGENT    1-64       3-212
DETCBS    1-72      17-44
DETCHK   17-66#     17-97
DETEMT    1-43      17-10#
DETERR   17-62#
DETGO    17-24#     17-96
DETHI    17-15      17-99#
DETKIL   17-76#     17-98
DETONE   17-33      17-53     17-61#    17-69     17-71     17-87
DETVEC   17-18      17-96#    17-99
DFJMEM    1-83       3-39
```

Cross reference table (CREF V05.05)

```
DIEARG   1-104
DIEMSG   1-104
DOTRMP   1-51     21-81*    21-88*    24-103*
DSPLIT  18-30     23-39#
ECO      1-25#
EC1      1-26#    15-24     16-25     17-61     19-59     22-27     24-34     27-20     33-16
EC10.    1-35#
EC2      1-27#    16-48     17-84     21-31     24-25     27-28
EC3      1-28#    16-79     24-17     26-16     28-39
EC4      1-29#    16-127    24-44     28-27
EC5      1-30#    16-150    22-35     28-33
EC6      1-31#    16-64     23-28     28-52
EC7      1-32#     3-222    27-48
EC8      1-33#    27-44
EC9      1-34#
EMSPHL   1-57      2-36
EMSPNH   1-57      2-37
EMTBLK   1-56      2-13      7-9       8-9       8-10      8-11      9-9       9-10      9-11      10-4      10-8      11-22
        12-4      13-4      15-17     16-30     16-62     16-86     16-101    16-116    16-130    16-148    16-155    17-14
        17-40     17-66     17-76     18-16     19-12     21-28     22-16     22-32     22-37     24-13     24-22     24-39
        26-26     26-38     26-46     26-54     27-25     27-34     28-14     32-14     33-31     33-33     33-35     33-45
        33-47     34-39     34-59     34-62
EMTLEV   1-58      3-9*
EMTXIT   1-67      7-11      8-22      9-22      10-23     11-51     12-14     13-9      14-7      15-20     16-169    17-59
        17-72     17-92     19-28     19-54     20-46     21-82     21-89     23-24     23-43     24-104    26-32     26-40
        26-48     26-56     28-81     28-120    29-16     29-32     33-66     34-6      34-13     34-82
ENQTL    1-70     11-17
ENSYS    1-82     33-55     34-57
ERRLOC   1-60      3-97*     3-117*
ESPLIT  18-29     23-19#
FORCEX   1-67     10-19     13-52
FP$MOV   1-82     33-56     34-58
FSTDL    1-71     17-31
FSTIOL   1-76     28-35
FSTSL    1-53     16-69
GETQ     1-52     16-95
GETUCH   1-72     16-106    17-45
GTUKBL   1-57      2-38
IB$IJ    1-52     16-118*
IB$SF2   1-52     16-96*    16-104
IDSFLG   1-84     23-19     23-39
ILSW2    1-77     12-7      28-67     28-92
INITLN   1-53     12-12     16-125    17-52
INTPRI   1-55      6-53
JSTK     1-63      3-204
JSWLOC   1-58      3-15*     3-172*
KILEMT   2-28     13-4#
KILJOB   1-43     13-8      13-19#    17-91
KMCPCX   1-57      2-35
KMNEMT   1-43      2-7#
KMNMAX   2-14      2-41#
KMNSUB   2-17      2-23#     2-41
KMRUN    2-23      3-7#
KMRVM    3-78      6-7#
KMSPDL   2-30     15-8#
KMSPEK   2-31      8-9#
```

Cross reference table (CREF V05.05)

```
KMSPLG   2-24     7-7#
KMSPOK   2-32     9-9#
KMSPRS   2-33    10-4#
KPAR0    1-66     6-49     6-50*    6-52*
KPAR6    1-82    33-59*   34-66*
LCLUNT   1-75     1-76    28-21    28-25    28-59*   28-82*
LCXTBL   1-52    28-60    28-89
LEMTPC   1-57     3-7*
LINIR    1-68    11-34*   28-66*   28-75*   28-84*
LINRTS   1-77    28-74    28-75
LINSWT   1-54    16-157
LJSW     1-58     3-14*    3-171*
LNPRIM   1-68    11-26    13-30    13-40    16-34    16-54    16-72    16-138   16-143*  35-27    35-27
LOGOFF   1-56     2-25
LOTSIZ   1-70    11-14
LOTSPC   1-70    11-14
LOUTIR   1-77    28-65*   28-74*   28-83*
LPARNT   1-72   16-139*   17-55*   35-29
LPROG    1-55    35-41    35-41
LPROJ    1-83    35-34    35-34
LQUAN    1-63     3-200*
LSECPT   1-53    16-35    16-55    16-140
LSTDL    1-71    17-32    17-68    17-80
LSTHL    1-75    28-17
LSTIOL   1-76    28-37
LSTPL    1-69    13-31    17-78    28-29    28-63
LSTSL    1-53    16-70    17-86
LSUCF    1-71    17-43
LSW      1-67    10-6     10-13*   10-18*   11-9     12-5     13-25    13-43*   13-44*   16-19    16-132*  17-34
        17-70    28-31
LSW10    1-78    12-13*   28-85*
LSW11    1-57     3-23*    3-177*   23-21*   23-41*
LSW2     1-61     3-139    16-17
LSW3     1-69    13-41*   13-42*   16-23    28-19    28-23    31-25
LSW4     1-56     2-7      3-8*
LSW6     1-54     3-183*
LSW9     1-61     3-131*   3-159    3-181    3-191    3-193*   3-210    16-21    16-129*  16-161*
LXCL     1-68    11-33*   11-43    13-33
MAPPLS   1-73    19-26
MAXSEC   1-53    16-36    16-45    16-47
MAXSRD   1-51    21-29    24-23    24-72
MXSFX   26-28    26-63#
NEDCHR   1-76    28-65
NEWJSW   1-59     3-13     3-72     3-82     3-83*    3-87*    3-129    3-170
NSPLDV   1-66     1-74     7-7     15-12    26-36    26-44    26-52
NUMWCB   1-50    24-15
ODTBAS   1-84    23-22
OVLBIT   1-59     3-72
OVRHC    1-67     7-10    11-39    12-12    13-35    15-18    16-125   16-133   16-157   17-52    19-26    26-39
        28-56    28-58    29-15    29-31    30-31
PO$BYP   1-70    27-42
PO$DET   1-71    17-24    17-82
P2$GRP   1-83    35-36
P2$SAM   1-83    35-43
P2$TRM   1-75    27-18
P2$WRL   1-83    35-21
```

Cross reference table (CREF V05.05)

```
PMBASE    1-81      33-32*    34-42     34-64
PMCELS    1-81       1-82     33-14     33-40     33-57
PMCHKU   34-4       34-11     34-33     34-89#
PMEMT     1-43      32-14#
PMEND    32-24      34-33#
PMFLGS    1-82      33-35*    34-48
PMINIT   32-21      33-14#
PMNBPC    1-82      33-48*    34-46     34-70
PMOVFR   34-67*     34-74*    34-79     34-83#
PMPAR     1-82      33-59     34-66
PMRUN     1-81      33-27*    34-5*     34-12*    34-34*
PMRVEC   32-19      32-21#
PMSTOP   32-23      34-11#
PMSTRT   32-22      34-4#
PMTOP     1-81      33-34*    34-44     34-71
PMUSER    1-81      33-18     33-26*    34-35*    34-89     34-92
PRGSIZ    1-58       3-27
PRGTOP    1-59       3-62
PRIVCO    1-71      17-24     17-82     27-42
PRIVC2    1-75      27-18     35-21     35-36     35-43
PS        1-23#      6-48*     6-53*
QNSPND    1-69      11-46
R$CHN     1-58       3-51
R$TTOP    1-62       3-154*
R$XCHN    1-59       3-53
R.GID     1-65       6-41
R.GSIZ    1-65       6-26      6-27*
R.GSTS    1-55       6-29*
RC$BAS    1-65       6-42      24-67
RC$DSP    1-50      24-47
RC$FLG    1-50      24-42      24-47
RC$LCG    1-50      24-42
RC$LEN    1-48      24-53
RCBBAS    1-50      25-11
RCBEND    1-50      25-13
RDB       1-73      19-40
RDBEND    1-73      19-41
REBOOT    1-56       2-27
RF$WRT    1-55      20-33     21-55
RMNBAS    1-60       1-62      3-128     3-154
RMON      1-61       3-133*    4-12*
RPAR      1-72      19-19*    20-23*
RPDR      1-72      19-20*    20-37*
RT$$SZ    1-73      19-47
RT$BAS    1-73      22-39     22-40
RT$FLG    1-55      20-33     21-55
RT$NAM    1-73      19-43     19-45
RT$TOP    1-73      22-33     22-44     22-46
RUNARG    1-60       3-103
RUNCHN    1-59       3-53
RUNRDB    1-65       6-27*     6-29*     6-30      6-41
S$CPU     1-70      11-16
S$SPND    1-69      11-45
S9600     1-77      12-9      28-70
SETDTR    1-54      11-39     28-58
SETERR    1-66       6-70     13-11     15-25     16-26     16-49     16-65     16-80     16-128    16-151    17-20     17-27
```

Cross reference table (CREF V05.05)

Cross reference table (CREF V05.05)

```
URO       1-72     16-13*    16-47*    16-165*   17-58*    22-50*    22-51*    24-59*    33-24*    34-94*
USDSPC    1-84     23-42
USRSTK    1-63      3-165
USTART    1-64      3-206
VALADB    1-71     16-88     17-42
VALADW    1-73     19-32     34-41     34-61
VIDCSR    1-64      4-22
VIMAGE    1-61      3-129
VLSEMT    1-44     16-13#
VO$RDB    1-65      6-25
VO$WDB    1-65      6-37
VO$WDE    1-65      6-39
VPAR6     1-82     33-58     34-65
VPLAS     1-72     19-24
VSWPFL    1-83      3-28
W.NOFF    1-66      6-46
W.NRID    1-66      6-44*
W.SIZE    1-66      6-54
WC$$SZ    1-49     24-30
WC$LEN    1-48     24-58*
WC$NPR    1-48     24-63*
WC$OFF    1-49     24-54     24-68
WC$PAR    1-49     24-69
WC$SIZ    1-49     24-32
WC$TRP    1-49     24-71*
WCBBAS    1-49     24-31
```

Cross reference table (CREF V05.05)