

2-	1	TSKMON EMT's
3-	1	KMRUN -- Start execution of a SAV file
7-	1	Misc. Kmon EMTs
16-	1	VLSEMT -- EMT to switch between subprocesses
17-	1	Detached job control emt's
18-	1	Shared Run-time control emt's
19-	1	Associate shared run-time system with job
20-	1	Map shared run-time system into user's virtual address
21-	1	Define a run-time region for fast mapping
23-	1	ESPLIT -- Enable user D-space
24-	1	Define a PLAS region for fast mapping
25-	1	CHKRCB -- Check address of Region Control Block
26-	1	Spooling control
27-	1	CL line control EMT's
28-	1	CLASN -- Assign a CL unit to a line
29-	1	CLCLR -- Clear XOFF sent and send XON
29-	18	CLRES -- Reset CL line
30-	1	CLCLOS -- Close channel opened to CL unit
31-	1	CLWAIT -- Wait for CL output to finish
32-	1	Performance monitor emt's
35-	1	CKACJB -- See if access to another job is allowed

```

1          .TITLE  TSEM4   TSX-Plus EMT Overlay
2          .ENABL  LC
3          .ENABL  AMA
4          .DSABL  GBL
5
6          ; Copyright (C) 1976, 1977, 1978, 1979, 1980, 1981, 1982, 1983, 1984, 1985, 1986,
7          ;          1987, 1988, 1989.
8
9          ; S&H Computer Systems, Inc.
10         ; Nashville, Tennessee
11
12         ; This software is furnished under a license for use only
13         ; on a single computer system and may be copied only with
14         ; the inclusion of the above copyright notice. This
15         ; software, or any other copies thereof, may not be provided
16         ; or otherwise made available to any other person except
17         ; for use on such system and to one who agrees to these
18         ; license terms. Title to and ownership of the software
19         ; shall at all times remain with S&H Computer Systems, Inc.
20
21         .CSECT  TSEM4
22 000000 TSEM4: .RAD50  /EM4/          ;Overlay id
23 000000 177776 PS          =          177776 ;Processor Status Word
24
25         ;
26         EC0      = 0          ;EMI error code return values
27         EC1      = 1
28         EC2      = 2
29         EC3      = 3
30         EC4      = 4
31         EC5      = 5
32         EC6      = 6
33         EC7      = 7
34         EC8      = 8
35         EC9      = 9
36         EC10     = 10
37
38         ; Macro calls
39         .MCALL   .READW, .CLOSE, .CRRG
40
41         ; Global definitions
42
43         .GLOBL  KMNEMT, DETEMT, PMEMT, SPLEMT, SSEMT, KILJOB
44         .GLOBL  CLEMT, CLCLOS, VLSEMT
45
46         ; Global references
47
48         .GLOBL  WC$LEN, WC$NPR, RC$LEN
49         .GLOBL  WC$$SZ, WCBBAS, WC$SIZ, WC$OFF, WC$PAR, WC$TRP, SR$WCB
50         .GLOBL  NUMWCB, RCBBAS, RCBEND, SHRRCB, SHRRCN, RC$FLG, RC$DSP, RC$LCG
51         .GLOBL  MAXSRD, SR$PAR, SR$PDR, SR$PX, SR$FLG, SR. MOR, SR. DSP, DOTRMP
52         .GLOBL  $VBELL, IB$SF2, SF2LEN, GETQ, IB$IJ, SPPRED, $VNOTT, LCXTBL
53         .GLOBL  $NOVLN, $SUCF, $LDFCF, $NOIN, MAXSEC, LSECPT, FSTSL, LSTSL, INITLN
54         .GLOBL  LSW6, CSHALC, CSHINI, SPLDEL, SETDTR, CD$DTR, CLWTIM, $RBRK, LINSWT
55         .GLOBL  $IOMAP, R. GSTS, INTPRI, CURRDB, UPARO, RF$WRT, RT$FLG, LPROG
56         .GLOBL  $INKMN, LSW4, BADEMT, EMTBLK, LOGOFF, SETHAN, REBOOT
57         .GLOBL  KMPCPX, EMSPHL, EMSPNH, QTUKBL, LEMTPC, CLREST, $RDSAV, LSW11

```

```

58 .GLOBL EMTLEV, LJSW, JSWLOC, PRGSIZ, SUTOP, CXTRMN, R#CHN, CLCLER
59 .GLOBL CHNSIZ, R#XCHN, RUNCHN, PRGTOP, CSIARE, DVLBIT, NEWJSW
60 .GLOBL CHAIN, CINFLG, UERSEV, ERRLOC, RUNARG, CINDAT, CPLEMT, RMNBAS
61 .GLOBL VIMAGE, $VIRJB, LSW9, RMON, LSW2, $SCOPE, TO$SCP, $TAB
62 .GLOBL TO$TAB, $FORM, TO$FF, TO$LC, CXTRMN, R$TTOP, TTOPTS, RMNBAS
63 .GLOBL $SETRN, USRSTK, $RNIOF, $IOMAP, SETMAP, $RNMLK, LQUAN, JSTK
64 .GLOBL UMODE, UPMODE, USTART, $DEBUG, DBGENT, ABORT, SUPRTN, VIDCSR
65 .GLOBL VO$RDB, R. $SIZ, RUNRDB, VO$WDB, VO$WDE, R. $ID, RC$BAS
66 .GLOBL W. NRID, W. NOFF, KPARO, W. SIZE, SETERR, NSPLDV
67 .GLOBL OVRHC, SPOLGO, EMTXIT, $DILUP, LSW, $SUSPN, FORCEX, $DETCH
68 .GLOBL LNPRIM, CL$XLN, CM$BRK, CM$ON, CL$STA, LXCL, CLXICP, LINIR
69 .GLOBL S$SPND, QNSPND, LSTPL, CLXBRK, $CTRLS, LSW3, $CTRLO
70 .GLOBL ENQTL, S$CPU, LOTSPC, LOTSIZ, PO$BYP
71 .GLOBL $DISCN, TRNSTR, PO$DET, PRIVCO, FSTD, LSTD, VALADB, LSUCF
72 .GLOBL DETCBS, GETUCH, CORUSR, LPARNT, URO, RPAR, RPDR, VPLAS
73 .GLOBL MAPPLS, VALADW, RDB, RDBEND, RT$NAM, RT$$SZ, RT$BAS, RT$TOP
74 .GLOBL CUPARO, CUPDRO, UPDRO, CS$OPN, CHNADR, C. CSW, NSPLDV, SPLSHF
75 .GLOBL P2$TRM, PRIVC2, CLTOTL, LSTHL, $HARD, LCLUNT, $DEAD, SPLSFF
76 .GLOBL FSTIOL, LSTIOL, CL$LIX, CL$RQH, CL$WQH, LCLUNT, NEDCHR, SPLSWF
77 .GLOBL LOUTIR, TTINCP, $AUTO, ILSW2, S9600, SETSPD, LINRTS, CLOTIR, CLINCP
78 .GLOBL $SXON, LSW10, CO$DEF, $TAB, CO$TAB, CO$FF, $8BIT, CO$8BT
79 .GLOBL CL$OPT, CL$STA, CL$COL, CL$LEN, CL$LIN, CL$SKP, CL$WID
80 .GLOBL CL$EPN, CL$EPS, CL$EPP, CS$SPL, CS$NMX, CLDEVX, C. DEVQ, C1DEVX
81 .GLOBL CL$ORS, CL$ORA, CM$EFP, $XCHAR, PMCELS, PMUSER, PMRUN, PMBASE, PMTOP
82 .GLOBL PMFLGS, PMCELS, PMNBPC, ENSYS, FP$MOV, VPAR6, PMPAR, KPAR6
83 .GLOBL P2$WRL, LPROJ, P2$GRP, P2$SAM, VSWPFL, DFJMEM
84 .GLOBL $UDSPC, ODTBAS, SR3MMR, USDSPC, IDSFLG
85 .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.

```

```
115  
116 . MACRO OCALL ENTADD  
117 . IF B, ENTADD  
118 . ERROR ; OCALL SPECIFIED WITH NO ENTRY ADDRESS  
119 . MEXIT  
120 . ENDC  
121 CALL OVRHC ; CALL THE OVERLAY HANDLER  
122 . WORD ENTADD ; SPECIFY THE ENTRY POINT  
123 . ENDM
```

```

1          .SBTTL  TSKMON EMT's
2          ;-----
3          ; KMNEMT is jumped to when a KMON specific EMT is executed.
4          ; It interprets the sub-function (in 2nd word of arg block)
5          ; and branches off to the appropriate processing routine.
6          ;
7 000002 032761 000000G 000000G KMNEMT: BIT    #$INKMN,LSW4(R1); IS KMON RUNNING NOW?
8 000010 001002                BNE    1$          ;BR IF YES
9 000012 000137 000000G        2$:   JMP    BADEMT    ;ILLEGAL EMT IF KMON NOT RUNNING
10         ;
11         ; Get sub-function code and branch off to processing routine.
12         ;
13 000016 013702 000002G        1$:   MOV    EMTBLK+2,R2    ;GET SUB-FUNCTION CODE
14 000022 020227 000022        CMP    R2,#KMNMAX    ;VALID SUB-FUNCTION CODE?
15 000026 103371                BHIS   2$          ;ERROR IF NOT
16 000030 006302                ASL    R2          ;CONVERT TO WORD TABLE INDEX
17 000032 000172 000036'        JMP    @KMNSUB(R2)    ;ENTER PROCESSING ROUTINE
18
19         ;-----
20         ; Table of KMON subfunctions for EMT code 126.
21         ; (Sub-function code is stored in 2nd word of EMT arg block)
22         ;
23 000036 000102' KMNSUB: .WORD  KMRUN    ; 00 Initiate execution of a SAV file
24 000040 001362' .WORD  KMSPLG   ; 01 Initiate a spooler
25 000042 000000G .WORD  LOGOFF   ; 02 Log off the current job
26 000044 000000G .WORD  SETHAN   ; 03 Update running copy of handler
27 000046 000000G .WORD  REBOOT   ; 04 Reboot the system
28 000050 001752' .WORD  KILEMT   ; 05 Force logoff of a job
29 000052 002122' .WORD  CSHNEW   ; 06 Clean out data cache
30 000054 002140' .WORD  KMSPDL   ; 07 Delete a spool file
31 000056 001406' .WORD  KMSPEK   ; 10 Get data from within system
32 000060 001436' .WORD  KMSPOK   ; 11 Store data into system
33 000062 001466' .WORD  KMSPRS   ; 12 Suspend or resume a job
34 000064 001540' .WORD  CLXCON   ; 13 Cross connect TT line to CL unit
35 000066 000000G .WORD  KMCPGX   ; 14 Copy data from context blk
36 000070 000000G .WORD  EMSPHL   ; 15 Set HOLD mode for a spooler
37 000072 000000G .WORD  EMSPNH   ; 16 Set NOHOLD mode for a spooler
38 000074 000000G .WORD  GTUKBL   ; 17 Access block in user key region
39 000076 001674' .WORD  STTTLN   ; 20 Start up a terminal line
40 000100 000000G .WORD  SETSTN   ; 21 Set site name for flag pages (in TSSPL2)
41         000022        KMNMAX = <-KMNSUB>/2
    
```

KMRUN -- Start execution of a SAV file

```

1          .SBTTL  KMRUN  -- Start execution of a SAV file
2          ;-----
3          ; KMRUN (126,0) is the EMT executed by KMON to start execution of
4          ; a SAV file.  On entry user channel # RUNCHN must be open to the SAV file.
5          ; The SAV file is read into memory and control is transfered to it.
6          ;
7 000102  005061  000000G  KMRUN:  CLR      LEMTPC(R1)      ;CLEAR ADDRESS OF LAST USER-MODE EMT
8 000106  042761  000000G  000000G  BIC      #KINKMN,LSW4(R1);SAY WE ARE EXITING FROM KMON
9 000114  012737  177777  000000G  MOV      #-1,EMTLEV    ;SAY NO EMT IS BEING DONE
10         ;
11         ; Set up JSW for job so virtual-image flag will be set before mem alloc
12         ;
13 000122  013746  000000G  MOV      NEWJSW,-(SP)   ;SET NEW JSW
14 000126  011661  000000G  MOV      (SP),LJSW(R1)
15 000132  106637  000000G  MTPD     @#JSWLOC
16         ;
17         ; Set $RDSAV flag to prevent PKSTAT/UPSTAT from saving/restoring the
18         ; JSW and Error byte in locations 44 and 52 of memory.  This is necessary
19         ; because SUTOP is going to change mapping to an uninitialized area
20         ; of memory and we don't want to use the contents of memory locations
21         ; until we actually get the SAV image in and set it up.
22         ;
23 000136  052761  000000G  000000G  BIS      #$RDSAV,LSW11(R1);Say we are reading in SAV file
24         ;
25         ; Set top of memory for the sav file we are about to start.
26         ;
27 000144  013700  000000G  MOV      PRGSIZ,RO     ;SET TOP OF MEMORY FOR JOB REGION
28 000150  105737  000000G  TSTB     VSWPFL        ;IS THIS A SWAPPING SYSTEM?
29         ; BNE      39$          ;IF SWAPPING USE JOB SIZE, ELSE
30         ; Changing this to always use PRGSIZ allows RT-11 utilities to
31         ; work correctly in a 64kb DFLMEM no swapping system.  They use
32         ; .GVAL offset 266 to determine base of USR (we set this in SUTOP
33         ; to 177774 if DFLMEM is 64kb) then TST -(RO) (which traps because
34         ; we only map PAR7 through the top of simulated RMON), prior to
35         ; doing a .SETTOP.  If the program is virtual (doesn't map RMON),
36         ; then PRGSIZ will be bigger than 56kb; if not, then pretend
37         ; that USR base is below RMON.  See TSKM2C CMDRUN for PRGSIZ set up.
38 000154  000406  BR        39$          ; Always use PRGSIZ as set up by TSKMON
39 000156  013700  000000G  MOV      DFJMEM,RO     ;RETRIEVE DEFAULT MEMORY SIZE
40 000162  000300  SWAB     RO            ;CONVERT TO ADDRESS ( * 256.)
41 000164  006300  ASL      RO            ; * 512.
42 000166  005300  DEC      RO            ;LEAVE 1 WORD SHORT (ELSE 64.KB = 0)
43 000170  006300  ASL      RO            ; * 1024. = ALLOCATED TOP ADDRESS IN BYTES
44 000172  004737  000000G  39$:    CALL     SUTOP    ;SET TOP OF MEMORY
45         ;
46         ; Transfer the channel information from internal channel RUNCHN
47         ; into user's channel # 17
48         ;
49 000176  013700  000000G  MOV      CXTRMN,RO     ;Get address of job's simulated RMON
50 000202  010003  MOV      RO,R3
51 000204  062703  000000G  ADD      #R#CHN+<17*CHNSIZ>,R3 ;Point to chan blk for channel 17
52 000210  010002  MOV      RO,R2
53 000212  062702  000000G  ADD      #R#XCHN+<<RUNCHN-21>*CHNSIZ>,R2 ;Point to RUNCHN block
54 000216  010200  MOV      R2,RO         ;Save pointer to RUNCHN block
55 000220  012704  000000G  MOV      #CHNSIZ/2,R4  ;Get # words to copy
56 000224  012223  4$:     MOV      (R2)+,(R3)+ ;Copy RUNCHN to channel 17
57 000226  077402  SOB      R4,4$

```

```

58 000230 005010          CLR      (R0)          ; Say RUNCHN is closed
59                      ;
60                      ; Read the SAV file into memory
61                      ;
62 000232 013703 000000G   MOV      PRGTOP,R3      ; GET HIGHEST ADDRESS IN SAV FILE IMAGE
63 000236 000241          CLC
64 000240 006003          RDR      R3              ; GET # WORDS TO READ
65 000242 005203          INC      R3              ; CVT FROM TOP ADDRESS TO # WORDS
66 000244          .READW  #CSIARE,#17,#0,R3,#0
67 000276 103002          BCC     16$            ; Br if ok
68 000300 000137 001000'   JMP     9$             ; Error during read
69                      ;
70                      ; If program is not overlaid, close channel 17.
71                      ;
72 000304 032737 000000G 000000G 16$: BIT     #OVLBIT,NEWJSW  ; IS THE PROGRAM OVERLAYED?
73 000312 001003          BNE     3$             ; IF YES THEN LEAVE CHANNEL 17 OPEN
74 000314          .CLOSE  #17          ; CLOSE CHANNEL 17
75                      ;
76                      ; See if we need to allocate extended-memory region for virtual overlays
77                      ;
78 000322 004737 001130'   3$:   CALL    KMRVM          ; See if we need to allocate XM region
79                      ;
80                      ; If a .CHAIN was done, move data from context block to chain data area.
81                      ;
82 000326 013703 000000G   MOV     NEWJSW,R3      ; GET JSW FOR JOB BEING STARTED
83 000332 042737 000000G 000000G   BIC     #CHAIN,NEWJSW ; CLEAR CHAIN FLAG
84 000340 105737 000000G   TSTB   CINFLG         ; WAS A .CHAIN DONE?
85 000344 001422          BEQ     20$            ; BR IF NOT
86 000346 105037 000000G   CLRB   CINFLG         ; SAY CHAIN IS FINISHED
87 000352 052737 000000G 000000G   BIS     #CHAIN,NEWJSW ; SET CHAIN-FLAG
88 000360 032703 000000G   BIT     #CHAIN,R3     ; DOES JOB WANT TO PRESERVE CHAIN DATA AREA?
89 000364 001034          BNE     1$             ; BR IF YES
90 000366 012704 000140   MOV     #<<1000-500>>/2,R4 ; GET # WORDS TO MOVE
91 000372 004737 001100'   CALL    CINMOV        ; Move chain data
92                      ;
93                      ; If a .CHAIN was done, reset the previous user's error cell (52&53).
94                      ;
95 000376 113746 000000G   MOVB   UERSEV,-(SP)   ; GET THE PREVIOUS ERROR SEVERITY LEVEL
96 000402 000316          SWAB   (SP)           ; PUT INTO HIGH BYTE
97 000404 106637 000000G   MTPD   @#ERRLOC      ; RESTORE THE USER ERROR SEVERITY LEVEL
98 000410 000425          BR     6$             ; SKIP THE INSTRUCTIONS TO CLEAR ERRLOC
99                      ;
100                     ; We are not doing a .CHAIN.
101                     ; See if we need to pass run argument string through chain data area.
102                     ;
103 000412 105737 000000G   20$:  TSTB   RUNARG      ; See if we need to pass RUN command line
104 000416 001005          BNE     22$            ; Br if yes
105 000420 012737 000001 000010G   MOV     #1,CINDAT+10  ; Say argument length = 1
106 000426 005037 000012G   CLR     CINDAT+12     ; Say argument string is null
107 000432 032703 000000G   22$:  BIT     #CHAIN,R3   ; Does job want to preserve chain data area?
108 000436 001007          BNE     1$             ; Br if yes
109 000440 013704 000010G   MOV     CINDAT+10,R4  ; Get # bytes in run argument string
110 000444 062704 000013   ADD     #13,R4        ; Get # bytes below 512 and round up
111 000450 006204          ASR     R4            ; Get # words to move
112 000452 004737 001100'   CALL    CINMOV        ; Move run argument to chain data area
113                     ;
114                     ; Initially clear user's error cell (52&53).

```

KMRUN -- Start execution of a SAV file

```

115 ;
116 000456 005046 1$: CLR -(SP) ;
117 000460 106637 0000000 MTPD @#ERRLOC ;INITIALLY ZERO USER'S ERROR CELL
118 ;
119 ; Store [EMT 356] instruction in job's low memory area to be used
120 ; to exit from a completion routine.
121 ;
122 000464 012746 104356 6$: MOV #104356, -(SP) ; [EMT 356]
123 000470 006637 0000000 MTPD @#CPLEMT ; STORE INTO JOB'S AREA
124 ;
125 ; Set pointer to simulated monitor vector table that is mapped through
126 ; user page 7 (160000-177777).
127 ;
128 000474 012746 0000000 MOV #RMNBAS, -(SP) ; VIRTUAL ADDRESS OF MONVEC TABLE
129 000500 032737 0000000 0000000 BIT #VIMAGE, NEWJSW ; IS THIS A VIRTUAL JOB?
130 000506 001405 BEQ 14$ ; BR IF NOT
131 000510 052761 0000000 0000000 BIS ##$VIRJB, LSW9(R1) ; SET VIRTUAL-JOB FLAG FOR THE SYSTEM
132 000516 012716 177776 MOV #177776, (SP) ; SET THIS AS RMON BASE FOR VIRTUAL JOB
133 000522 106637 0000000 14$: MTPD @#RMON ; SET POINTER IN USER'S SPACE
134 ;
135 ; Set EMT vector cell (location 30) in user's area to point to
136 ; cell in simulated RMON vector.
137 ;
138 000526 005000 CLR R0 ; INITIALIZE OPTION WORD VALUE
139 000530 016102 0000000 MOV LSW2(R1), R2 ; GET VALUE OF LSW2 FLAGS FOR LINE
140 000534 032702 0000000 BIT ##$SCOPE, R2 ; IS THIS A SCOPE TYPE TERMINAL?
141 000540 001402 BEQ 10$ ; BR IF NOT
142 000542 052700 0000000 BIS #TO$SCP, R0 ; SET SCOPE FLAG
143 000546 032702 0000000 10$: BIT ##$TAB, R2 ; HARDWARE TABS?
144 000552 001402 BEQ 11$ ; BR IF NOT
145 000554 052700 0000000 BIS #TO$TAB, R0 ; SET TAB FLAG
146 000560 032702 0000000 11$: BIT ##$FORM, R2 ; HARDWARE FF?
147 000564 001402 BEQ 12$ ; BR IF NOT
148 000566 052700 0000000 BIS #TO$FF, R0 ; SET FORM-FEED FLAG
149 000572 032702 0000000 12$: BIT #TO$LC, R2 ; ALLOW LOWER-CASE INPUT?
150 000576 001402 BEQ 13$ ; BR IF NOT
151 000600 052700 0000000 BIS #TO$LC, R0 ; SET LC FLAG
152 000604 013702 0000000 13$: MOV CXTRMN, R2 ; GET ADDRESS OF BASE OF SIMULATED RMON FOR JOB
153 000610 010062 0000000 MOV R0, R#$TTOP(R2) ; SET TT OPTION WORD VALUE
154 000614 012746 0000000 MOV #TTOPTS+RMNBAS+2, -(SP) ; GET ADDRESS OF TT OPTION CELL
155 000620 106637 0000030 MTPD @#30 ; SET IN EMT TRAP VECTOR
156 ;
157 ; See if we need to some special initialization for SETUP program
158 ;
159 000624 032761 0000000 0000000 BIT ##$SETRN, LSW9(R1) ; Is this the SETUP program being started?
160 000632 001402 BEQ 19$ ; Br if not
161 000634 004737 001024' CALL SETINI ; Initialize for SETUP program
162 ;
163 ; Set up user's SP
164 ;
165 000640 013746 0000000 19$: MOV USRSTK, -(SP) ; STACK POINTER FOR PROGRAM BEING STARTED
166 000644 106606 MTPD SP ; SET AS USER-MODE SP
167 ;
168 ; Set up JSW for job in memory now that SAV image has been read in
169 ;
170 000646 013746 0000000 MOV NEWJSW, -(SP) ; SET NEW JSW
171 000652 011661 0000000 MOV (SP), LJSW(R1)

```


KMRUN -- Start execution of a SAV file

```

172 000656 106637 000000G          MTPD   @#JSWLOC
173                               ;
174                               ; Now that memory image is set up correctly, clear the flag that says
175                               ; we are reading in the SAV file.
176                               ;
177 000662 042761 000000G 000000G          BIC   #RDSAV,LSW11(R1);Finished reading in SAV file
178                               ;
179                               ; See if we should map PAR 7 for job to I/O page
180                               ;
181 000670 032761 000000G 000000G          BIT   #RNIOP,LSW9(R1);Is I/O page access wanted?
182 000676 001403                               BEQ   17$           ;Br if not
183 000700 052761 000000G 000000G          BIS   #IOMAP,LSW6(R1);Set flag saying to map PAR 7 to I/O page
184                               ;
185                               ; Make sure memory management is set up correctly for the job
186                               ;
187 000706 004737 000000G          17$:  CALL   SETMAP           ;Set up memory mapping for the job
188                               ;
189                               ; See if we need to lock this program in low memory
190                               ;
191 000712 032761 000000G 000000G          BIT   #RNLK,LSW9(R1);Should we lock this program in memory?
192 000720 001410                               BEQ   18$           ;Br if not
193 000722 042761 000000G 000000G          BIC   #RNLK,LSW9(R1);Clear the flag
194 000730 012700 000000G          MOV   #CSIARE,R0      ;Point to EMT arg block area
195 000734 012710 060007          MOV   #<7+<140*400>>, (R0) ;Set up EMT argument block
196 000740 104375          EMT   375           ;Try to lock job in memory
197                               ;
198                               ; Give job a full time-slice
199                               ;
200 000742 005061 000000G          18$:  CLR   LQUAN(R1)      ;GIVE FULL TIME-SLICE ON STARTUP
201                               ;
202                               ; Push starting PSW and PC onto stack
203                               ;
204 000746 012706 000000G          MOV   #JSTK,SP        ;CLEAN OFF CONTEXT-BLOCK STACK
205 000752 012746 000000G          MOV   #UMODE!UPMODE,-(SP);USER-MODE PS
206 000756 013746 000000G          MOV   USTART,-(SP)    ;STARTING ADDRESS
207                               ;
208                               ; See if we are to start program under the debugger
209                               ;
210 000762 032761 000000G 000000G          BIT   #DEBUG,LSW9(R1);Are we to run program under the debugger?
211 000770 001402                               BEQ   15$           ;Br if not
212 000772 000137 000000G          JMP   DBGENT          ;Enter debugger to start the program
213                               ;
214                               ; Use RTI to enter user's program in user mode.
215                               ;
216 000776 000002          15$:  RTI                   ;ENTER PROGRAM IN USER-MODE
217                               ;
218                               ; Error reading SAV file
219                               ;
220 001000 105037 000000G          9$:   CLRB   CINFLG      ;Say chain is not being done
221 001004          .CLOSE   #17          ;Close the SAV file channel
222 001012 012705 000007          MOV   #EC7,R5        ;Get error code
223 001016 005004          CLR   R4
224 001020 000137 000000G          JMP   ABORT          ;Abort the job

```

KMRUN -- Start execution of a SAV file

```

1          ; -----
2          ; Perform special initialization when starting execution of SETUP program.
3          ; We must do this because SETUP tries to call the GETCSR routine to
4          ; get the CSR address of the video controller when running on the PRO.
5          ;
6 001024   SETINI:
7          ;
8          ; Set up a special RMON pointer for SETUP which will cause offset
9          ; 434 to point to a routine which will return video CSR address.
10         ;
11 001024   012746   177344G   MOV     #SUPRTN-434,-(SP);Set fake address for RMON pointer
12 001030   106637   000000G   MTPD   @#RMON           ;Store RMON pointer into job space
13 001034   012700   000000G   MOV     #SUPRTN,RO      ;Get location where we store code in job
14 001040   012746   000002G   MOV     #SUPRTN+2,-(SP);Set address where code will be stored
15 001044   106620           MTPD   (RO)+           ;Store into job space
16         ;
17         ; Now store code in job space which will return CSR address for
18         ; video controller on the top of the stack.
19         ;
20 001046   012746   012766   MOV     #012766,-(SP)   ; [MOV #csr,2(SP)]
21 001052   006620           MTPD   (RO)+           ;Store into program space
22 001054   013746   000000G   MOV     VIDCSR,-(SP)   ;Get CSR address for video controller
23 001060   006620           MTPD   (RO)+           ;Store into program space
24 001062   012746   000002G   MOV     #2,-(SP)
25 001066   006620           MTPD   (RO)+           ;Store stack offset for MOV instruction
26 001070   012746   000207   MOV     #207,-(SP)     ; [RETURN]
27 001074   006620           MTPD   (RO)+           ;Store into program space
28         ;
29         ; Finished
30         ;
31 001076   000207           RETURN

```

KMRUN -- Start execution of a SAV file

```
1 ;-----  
2 ; Move chain data from job context area to locations starting at 500  
3 ; in the memory image of the program being started.  
4 ;  
5 ; Inputs:  
6 ; R4 = Number of words to move.  
7 ;  
8 001100 010346 CINMOV: MOV R3, -(SP)  
9 001102 010446 MOV R4, -(SP)  
10 ;  
11 ; Move the data  
12 ;  
13 001104 012700 000500 MOV #500, R0 ;Get address of area to receive data  
14 001110 012703 000000G MOV #CINDAT, R3 ;Point to chain data area in context block  
15 001114 012346 2$: MOV (R3)+, -(SP) ;Move chain data from context block to job  
16 001116 106620 MTPD (R0)+  
17 001120 077403 SOB R4, 2$  
18 ;  
19 ; Finished  
20 ;  
21 001122 012604 MOV (SP)+, R4  
22 001124 012603 MOV (SP)+, R3  
23 001126 000207 RETURN
```

KMRUN -- Start execution of a SAV file

```

1          ; -----
2          ; KMRVM is called from KMRUN when setting up a job for execution.
3          ; It checks to see if the job requires an extended-memory region for
4          ; virtual overlays, and if so allocates the region and zeroes the
5          ; segment id values in each section of the region.
6          ;
7 001130   010146   KMRVM:  MOV     R1,-(SP)
8 001132   010246           MOV     R2,-(SP)
9 001134   010346           MOV     R3,-(SP)
10 001136   010446          MOV     R4,-(SP)
11 001140   010546          MOV     R5,-(SP)
12          ;
13          ; Determine if this job needs an extended memory region
14          ;
15 001142   106537   000066   MFPGD   @#66           ;Get pointer to Window Definition Blocks
16 001146   022627   001000   CMP     (SP)+,#1000   ;Any windows for job?
17 001152   103471           BLD     9#            ;Br if not -- No XM region needed
18          ;
19          ; Create an extended memory region using the information in the
20          ; Region Definition Block in the job.
21          ;
22 001154   106537   000064   MFPGD   @#64           ;Get pointer to job's overlay table
23 001160   012601           MOV     (SP)+,R1
24 001162   010102           MOV     R1,R2         ;Point to region definition block
25 001164   062702   000000G   ADD     #VO#RDB,R2
26 001170   106562   000000G   MFPGD   R,GSIZ(R2)    ;Get # 64-byte blocks to allocate for region
27 001174   012637   000000C   MOV     (SP)+,RUNRDB+R,GSIZ
28 001200   001456           BEQ     9#            ;Br if no space needed for XM region
29 001202   005037   000000C   CLR     RUNRDB+R,GSTS ;Clear Region Definition Block status flags
30 001206           .CRRG   #CSIARE,#RUNRDB ;Create the XM region
31 001226   103451           BCS    10#           ;Br if error while creating the region
32          ;
33          ; Zero the ID numbers in each section of the region.
34          ; This is done to indicate that the code for the section needs to be
35          ; read into the region from disk.
36          ;
37 001230   106561   000000G   MFPGD   VO#WDB(R1)    ;Get pointer to first Window Definition Block
38 001234   012602           MOV     (SP)+,R2
39 001236   106561   000000G   MFPGD   VO#WDE(R1)    ;Get pointer past end of last WDB
40 001242   012603           MOV     (SP)+,R3
41 001244   013704   000000C   MOV     RUNRDB+R,GID,R4 ;Get address of system Region Control Block
42 001250   016400   000000G   MOV     RC#BAS(R4),R0 ;Get base 64-byte page number of region base
43 001254   010446           1#:  MOV     R4,-(SP)    ;Store pointer to Region Control Block in WDB
44 001256   106662   000000G   MTPD   W.NRID(R2)
45 001262   010005           MOV     R0,R5         ;Get base page number of region
46 001264   106562   000000G   MFPGD   W.NOFF(R2)    ;Get offset of window within region
47 001270   062605           ADD     (SP)+,R5      ;Get base page # of window
48 001272           DISABL           ;: DISABLE
49 001300   013746   000000G   MOV     @#KPAR0,-(SP) ;: Save mapping for kernel page 0
50 001304   010537   000000G   MOV     R5,@#KPAR0    ;: Map kernel page 0 to window section
51 001310   005037   000000   CLR     @#0           ;: Zero section ID number
52 001314   012637   000000G   MOV     (SP)+,@#KPAR0 ;: Restore system par 0 mapping
53 001320           ENABL           ;: ENABLE
54 001326   062702   000000G   ADD     #W.SIZE,R2    ;Point to next window definition block
55 001332   020203           CMP     R2,R3         ;Have we done all windows?
56 001334   103747           BLD     1#            ;Loop if more windows to do
57          ;

```

```
58 ; Finished
59 ;
60 001336 012605 9#: MOV (SP)+,R5
61 001340 012604 MOV (SP)+,R4
62 001342 012603 MOV (SP)+,R3
63 001344 012602 MOV (SP)+,R2
64 001346 012601 MOV (SP)+,R1
65 001350 000207 RETURN
66 ;
67 ; Error -- Unable to create XM region for job
68 ;
69 001352 012700 177754 10#: MOV #-24,R0 ;Return error code
70 001356 000137 0000000 JMP SETERR
```

```
1          .SBTTL  Misc. Kmon EMTs
2          ;-----
3          ; SPOLGD (1) Initiate operation of a spooler.
4          ; Argument word 2 contains the address of the SDCB for the spooled device
5          ; being initiated.
6          ;
7 001362 105737 000000G  KMSPLG: TSTB   NSPLDV           ;Are there any spooled devices?
8 001366 001405          BEQ     9$           ;BR IF NOT
9 001370 013700 000004G      MOV     EMTBLK+4,R0      ;GET ADDRESS OF SDCB
10 001374          OCALL  SPOLGD           ;START THE SPOOLER
11 001402 000137 000000G  9$:   JMP     EMTXIT          ;FINISHED
```

```
1 ;-----  
2 ; Get data from within the system.  
3 ;  
4 ; Argument words in EMT block:  
5 ; Word 2: Address of start of data within the kernel.  
6 ; Word 3: Number of bytes of data to transfer.  
7 ; Word 4: Address of buffer within TSKMON where data is to be placed.  
8 ;  
9 001406 013702 0000040 KMSPEK: MOV EMTBLK+4,R2 ;Get address of data wanted  
10 001412 013703 0000060 MOV EMTBLK+6,R3 ;Get # bytes to transfer  
11 001416 013704 0000100 MOV EMTBLK+10,R4 ;Get address of buffer in TSKMON  
12 ;  
13 ; Move data from kernel to TSKMON buffer  
14 ;  
15 001422 006203 ASR R3 ;Get # words to transfer  
16 001424 012246 1$: MOV (R2)+,-(SP) ;Get a word of data onto the stack  
17 001426 106624 MTPD (R4)+ ;Store into TKMON buffer  
18 001430 077303 SOB R3,1$ ;Loop till all words moved  
19 ;  
20 ; Finished  
21 ;  
22 001432 000137 0000000 JMP EMTXIT ;Finished
```

```
1 ;-----  
2 ; Store data into kernel data structure.  
3 ;  
4 ; Argument words in EMT block:  
5 ; Word 2: Address of start of data within the kernel.  
6 ; Word 3: Number of bytes of data to transfer.  
7 ; Word 4: Address of buffer within TSKMON where data is to be fetched.  
8 ;  
9 001436 013702 0000040 KMSPOK: MOV EMTBLK+4,R2 ;Get address of data destination  
10 001442 013703 0000060 MOV EMTBLK+6,R3 ;Get # bytes to transfer  
11 001446 013704 0000100 MOV EMTBLK+10,R4 ;Get address of buffer in TSKMON  
12 ;  
13 ; Move data from TSKMON buffer to kernel  
14 ;  
15 001452 006203 ASR R3 ;Get # words to transfer  
16 001454 106524 1$: MFPD (R4)+ ;Get a word of data from TSKMON buffer  
17 001456 012622 MOV (SP)+,(R2)+ ;Store into kernel space  
18 001460 077303 SOB R3,1$ ;Loop till all words moved  
19 ;  
20 ; Finished  
21 ;  
22 001462 000137 0000000 JMP EMTXIT ;Finished
```



```

1 ;-----
2 ; Suspend or resume execution of a job.
3 ;
4 001466 013701 0000040 KMSPRS: MOV EMTBLK+4,R1 ;Get # of job to suspend or resume
5 001472 001420 BEQ 9$ ;Zero is invalid
6 001474 032761 0000000 0000000 BIT ##DILUP,LSW(R1) ;Is job logged on?
7 001502 001414 BEQ 9$ ;Br if not
8 001504 105737 0000000 TSTB EMTBLK ;Suspend or resume the job?
9 001510 001004 BNE 1$ ;Br if resume
10 ;
11 ; Suspend execution of the job
12 ;
13 001512 052761 0000000 0000000 BIS ##SUSPN,LSW(R1) ;Set suspend-execution flag for job
14 001520 000405 BR 9$
15 ;
16 ; Resume execution of the job
17 ;
18 001522 042761 0000000 0000000 1$: BIC ##SUSPN,LSW(R1) ;Clear suspend-job flag
19 001530 004737 0000000 CALL FORCEX ;Force job to run
20 ;
21 ; Finished
22 ;
23 001534 000137 0000000 9$: JMP EMTXIT
  
```

```

1 ; -----
2 ; Cross connect a TT line to a CL unit.
3 ; Word 3 of EMT argument block contains CL unit number.
4 ;
5 001540 CLXCON:
6 ;
7 ; Treat this as a NOP if this is a detached job
8 ;
9 001540 032761 0000000 0000000 BIT    #$DETCH,LSW(R1) ;Is this a detached job?
10 001546 001050 BNE    9$           ;Br if yes
11 ;
12 ; Ensure that pending terminal output is flushed
13 ;
14 001550 026161 0000000 0000000 2$:    CMP    LOTSPC(R1),LOTSIZ(R1) ;Any output still pending?
15 001556 001405 BEQ    3$           ;Go ahead if not
16 001560 012700 0000000 MOV    #$CPU,R0     ;Get out of everybody's way while waiting
17 001564 004737 0000000 CALL   ENQTL        ;Give somebody else a chance while waiting
18 001570 000767 BR     2$           ;And check again
19 ;
20 ; Get Desired CL unit
21 ;
22 001572 013705 0000040 3$:    MOV    EMTBLK+4,R5     ;Get desired CL unit index number
23 ;
24 ; Say CL unit connected to TT line
25 ;
26 001576 016104 0000000 MOV    LNPRIM(R1),R4 ;Get primary line #
27 001602 010465 0000000 MOV    R4,CL$XLN(R5) ;Set # of TT line connected to CL
28 001606 042765 0000000 0000000 BIC    #^C<CM$BRK>,CL$STA(R5) ;Init status flags for CL unit
29 001614 052765 0000000 0000000 BIS    #CM$ON,CL$STA(R5) ;And say line is in use
30 ;
31 ; Connect TT unit to CL line
32 ;
33 001622 010564 0000000 MOV    R5,LXCL(R4)   ;Connect CL unit to our line
34 001626 012764 0000000 0000000 MOV    #CLXICP,LINIR(R4) ;Set addr of input char processing routine
35 ;
36 ; Raise DTR on CL unit
37 ;
38 001634 052765 0000000 0000000 BIS    #CO$DTR,CL$OPT(R5) ;Request DTR up
39 001642 OCALL SETDTR ;Call CL routine to raise DTR
40 ;
41 ; Suspend execution of job until cross connection is broken
42 ;
43 001650 005764 0000000 1$:    TST    LXCL(R4)     ;Are we still cross connected?
44 001654 002405 BLT    9$           ;Br if not
45 001656 012700 0000000 MOV    #$SPND,R0    ;Get sleep state
46 001662 004737 0000000 CALL   QNSPND       ;Suspend execution of job
47 001666 000770 BR     1$           ;See if connection has been broken
48 ;
49 ; Finished
50 ;
51 001670 000137 0000000 9$:    JMP    EMTXIT

```

```
1 ;-----  
2 ; Start a terminal line  
3 ;  
4 001674 013701 000004G STTTLN: MOV EMTBLK+4,R1 ;Get index # of line to be started  
5 001700 032761 000000G 000000G BIT ##DILUP,LSW(R1) ;Is line already started?  
6 001706 001017 BNE 9$ ;Br if yes  
7 001710 032761 000000G 000000G BIT ##AUTO,ILSW2(R1);Is line set to autobaud?  
8 001716 001404 BEQ 1$ ;Br if not  
9 001720 012700 000000G MOV #S9600,R0 ;Set speed to 9600 baud  
10 001724 004737 000000G CALL SETSPD ;Set the line speed  
11 001730 005000 1$: CLR R0 ;No secondary start-up command file  
12 001732 OCALL INITLN ;Initialize the line  
13 001740 042761 000000G 000000G BIC ##RBRK,LSW10(R1);Clear break-received flag  
14 001746 000137 000000G 9$: JMP EMTXIT
```

```

1 ;-----
2 ; Force logoff of a job.
3 ;
4 001752 013702 000004G KILEMT: MOV EMTBLK+4,R2 ;GET # OF JOB TO KILL
5 001756 004737 006650' CALL CKACJB ;See if we have privilege to kill that job
6 001762 103405 BCS 10$ ;Br if don't have privilege
7 001764 010200 MOV R2,R0 ;Get # of job to kill
8 001766 004737 002004' CALL KILJOB ;KILL THE JOB
9 001772 000137 000000G JMP EMTXIT ;FINISHED
10 001776 005000 10$: CLR R0 ;Return error code 0
11 002000 000137 000000G JMP SETERR
12 ;-----
13 ;
14 ; Subroutine to "kill" (force logoff) of a job.
15 ;
16 ; Inputs:
17 ; R0 = Virtual job index number of job to kill.
18 ;
19 002004 010146 KILJOB: MOV R1,-(SP)
20 002006 010546 MOV R5,-(SP)
21 002010 010001 MOV R0,R1 ;Get virtual job index number
22 ;
23 ; See if job is currently logged on
24 ;
25 002012 032761 000000G 000000G BIT #$DILUP,LSW(R1) ;IS JOB LOGGED ON NOW?
26 002020 001435 BEQ 9$ ;BR IF NOT
27 ;
28 ; If this line is cross connected with a CL line, break the connection.
29 ;
30 002022 016105 000000G MOV LNPRIM(R1),R5 ;Get primary line number
31 002026 020527 000000G CMP R5,#LSTPL ;Is this a primary line (not detached)?
32 002032 101006 BHI 1$ ;Br if detached job
33 002034 016505 000000G MOV LXCL(R5),R5 ;Is line cross connected to a CL unit?
34 002040 002403 BLT 1$ ;Br if not
35 002042 QCALL CLXBRK ;Break the connection
36 ;
37 ; Clear ctrl-S output suspension and ctrl-O output suppression and
38 ; set flag saying job is being killed.
39 ;
40 002050 016100 000000G 1$: MOV LNPRIM(R1),R0 ;GET PRIMARY LINE NUMBER
41 002054 042760 000000G 000000G BIC #$CTRLS,LSW(R0);RESET CTRL-S OUTPUT SUSPENSION
42 002062 042761 000000G 000000G BIC #$CTRL0,LSW(R1);RESET CTRL-O OUTPUT SUSPENSION
43 002070 042761 000000G 000000G BIC #$SUSPN,LSW(R1) ;Clear job-suspended flag
44 002076 052761 000000G 000000G BIS #$DISCN,LSW(R1) ;FORCE LOGOFF
45 ;
46 ; Start output transmission for job
47 ;
48 002104 004777 000000G CALL @TRNSTR ;Start output transmission
49 ;
50 ; Force job to execute
51 ;
52 002110 004737 000000G CALL FORCEX ;FORCE ITS EXECUTION
53 ;
54 ; Finished
55 ;
56 002114 012605 9$: MOV (SP)+,R5
57 002116 012601 MOV (SP)+,R1

```

58 002120 000207

RETURN

```
1 ;-----  
2 ; Clean out data cache.  
3 ;  
4 002122 005737 0000006 CSHNEW: TST CSHALC ;Is data caching genned into system?  
5 002126 001402 BEQ 1$ ;Br if not  
6 002130 004777 0000006 CALL @CSHINI ;Clean out the data cache  
7 002134 000137 0000006 1$: JMP EMTXIT
```

```
1 ;-----  
2 ; Delete a spool file.  
3 ; Form of the EMT:  
4 ; .BYTE 0,126  
5 ; .WORD 7  
6 ; .WORD file_id_number  
7 ;  
8 002140 KMSDDL:  
9 ;  
10 ; See if spooling support is included in this system  
11 ;  
12 002140 105737 0000000 TSTB NSPLDV ;Are there any spooled devices?  
13 002144 001410 BEQ 1$ ;Br if not  
14 ;  
15 ; Get the file id number and delete the file  
16 ;  
17 002146 013703 0000040 MOV EMTBLK+4,R3 ;Get spool file ID number  
18 002152 OCALL SPLDEL ;Try to delete the file  
19 002160 103402 BCS 1$ ;Br if could not find the file  
20 002162 000137 0000000 JMP EMTXIT  
21 ;  
22 ; Specified file could not be found  
23 ;  
24 002166 012700 000001 1$: MOV #EC1,R0  
25 002172 000137 0000000 JMP SETERR
```

VLSEMT -- EMT to switch between subprocesses

```

1          .SBTTTL  VLSEMT -- EMT to switch between subprocesses
2          ;-----
3          ; The form of the EMT used to switch between subprocesses is
4          ;
5          ; .BYTE   sub_function,162
6          ; .BYTE   subprocess_number,0
7          ; .BYTE   return_process,initiate_only
8          ; .WORD   pointer_to_start_up_file_string
9          ;
10         ; Where subprocess_number is the relative subprocess number for the
11         ; job (0=primary process).
12         ;
13 002176 005037 000000G VLSEMT: CLR      URO          ;Return 0 in RO if error
14         ;
15         ; See if job is authorized to use subprocesses
16         ;
17 002202 032761 000000G 000000G BIT      ##NOVLN,LSW2(R1);Is job allowed to use subprocesses?
18 002210 001014 BNE      1$          ;Br if not
19 002212 032761 000000G 000000G BIT      ##DETCH,LSW(R1) ;Is this a detached job?
20 002220 001010 BNE      1$          ;Br if detached
21 002222 032761 000000C 000000G BIT      #<#SUCF!#LDFCF>,LSW9(R1) ;Doing logon/logoff processing?
22 002230 001004 BNE      1$          ;Br if yes
23 002232 032761 000000G 000000G BIT      ##NOIN,LSW3(R1) ;Doing logon/logoff processing?
24 002240 001404 BEQ      2$          ;Br if not
25 002242 012700 000001 1$:      MOV      #EC1,RO      ;Error 1 if not allowed to use subprocesses
26 002246 000137 000000G JMP      SETERR
27         ;
28         ; See if he specified a process number or wants us to find a free one
29         ;
30 002252 113705 000002G 2$:      MOVVB   EMTBLK+2,R5      ;Get specified subprocess number
31 002256 001570 BEQ      4$          ;Br if switching to primary process
32 002260 003015 BGT      12$         ;Br if subprocess number specified
33 002262 012705 000001 MOV      #1,R5      ;Start with subprocess # 1
34 002266 016103 000000G MOV      LNPRIM(R1),R3 ;Get primary line #
35 002272 016303 000000G MOV      LSECPT(R3),R3 ;Point to table of secondary line numbers
36 002276 020527 000000G 13$:   CMP      R5,#MAXSEC ;Are there more subprocesses available to job?
37 002302 101050 BHI      6$          ;Br if not
38 002304 105723 TSTB   (R3)+        ;Is this subprocess free?
39 002306 001423 BEQ      15$         ;Br if free
40 002310 005205 INC      R5          ;Advance subprocess number
41 002312 000771 BR      13$         ;Continue searching for a free subprocess
42         ;
43         ; See if the specified process number is valid
44         ;
45 002314 020527 000000G 12$:   CMP      R5,#MAXSEC ;Is subprocess number too large?
46 002320 101407 BLOS   3$          ;Br if ok
47 002322 012737 000000G 000000G MOV      #MAXSEC,URO ;Return MAXSEC to user in RO
48 002330 012700 000002 MOV      #EC2,RO      ;Error 2 -- Subprocess number too big
49 002334 000137 000000G JMP      SETERR
50         ;
51         ; Switching to a subprocess.
52         ; See if subprocess has been allocated yet.
53         ;
54 002340 016102 000000G 3$:      MOV      LNPRIM(R1),R2 ;Get primary line #
55 002344 016203 000000G MOV      LSECPT(R2),R3 ;Point to table of secondary line numbers
56 002350 060503 ADD      R5,R3      ;Point to entry for selected process
57 002352 105743 TSTB   -(R3)        ;Has this subprocess been allocated yet?

```


VLSEMT -- EMT to switch between subprocesses

```

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

```

VLSEMT -- EMT to switch between subprocesses

```

115 002520 012601          MOV      (SP)+,R1      ;Get back current process index
116 002522 105737 000004G  TSTB    EMTBLK+4      ;Return to our process or primary process?
117 002526 001402          BEQ     10$           ;Br if return to primary process
118 002530 110160 000000G  MOVB   R1,IB$IJ(R0)   ;Set our process # as initiating job
119
120          ; At this point, R0=Pointer to buffer with initialization info (IB$xxx).
121          ; Initialize the subprocess.
122
123 002534 010146 10$:    MOV     R1,-(SP)      ;Save original process number
124 002536 010301          MOV     R3,R1         ;Get new process number
125 002540          OCALL  INITLN      ;Initialize the new process
126 002546 103004          BCC    8$            ;Br if initialization ok
127 002550 012700 000004   MOV     #EC4,R0       ;Error 4 -- Insufficient memory for process
128 002554 000137 000000G  JMP     SETERR
129 002560 052761 000000G 8$:    BIS     #$VBELL,LSW9(R1); Don't ring bell when new subprocess waits
130 002566 105737 000005G  TSTB    EMTBLK+5      ;Are we going to switch terminal control?
131 002572 001404          BEQ     20$           ;Br if yes
132 002574 052761 000000G 000000G  BIS     #$VNOTT,LSW(R1); Say subprocess not connected to terminal
133 002602          OCALL  SPPRED      ;Reduce execution priority of initiated proc
134 002610 012601 20$:    MOV     (SP)+,R1      ;Get back original process number
135
136          ; Make association between subprocess and primary process
137
138 002612 016102 000000G  MOV     LNPRIM(R1),R2  ;Get primary process number
139 002616 010263 000000G  MOV     R2,LPARNT(R3) ;Say primary job is parent of subprocess
140 002622 016204 000000G  MOV     LSECT(R2),R4   ;Point to subprocess # table
141 002626 060504          ADD     R5,R4         ;Point to desired entry
142 002630 110344          MOVB   R3,-(R4)      ;Associate subprocess with primary
143 002632 010263 000000G  MOV     R2,LNPRIM(R3) ;Set primary process for subprocess
144 002636 000410          BR     19$
145
146          ; We are switching to a process that has been initialized.
147
148 002640 123727 000000G 000001 4$:    CMPB   EMTBLK,#1      ;Can we use an existing process?
149 002646 001004          BNE    19$           ;Br if yes
150 002650 012700 000005   MOV     #EC5,R0       ;Error 5 -- Process already exists
151 002654 000137 000000G  JMP     SETERR
152
153          ; Switch terminal connection to the selected process.
154
155 002660 105737 000005G 19$:    TSTB   EMTBLK+5      ;Initiate subprocess only?
156 002664 001006          BNE    9$            ;Br if yes -- Don't switch terminal control
157 002666          OCALL  LINSWT      ;Set up line number prompt
158
159          ; Set flag to suppress bell ringing when this line goes into wait state
160
161 002674 052761 000000G 000000G  BIS     #$VBELL,LSW9(R1); Say not to ring bell when we wait
162
163          ; Return subprocess number to calling program in R0
164
165 002702 010537 000000G 9$:    MOV     R5,URO       ;Return subprocess # in R0
166
167          ; Finished
168
169 002706 000137 000000G  JMP     EMTXIT      ;Finished with EMT

```

```

1          .SBTTL Detached job control emt's
2          ;-----
3          ; This set of emt's is used to control detached jobs.
4          ; The emt function code is 132. The sub-function code is stored in
5          ; byte 0 of the argument block. The valid sub-function codes are:
6          ; 0==>Start a detached job.
7          ; 1==>Check status of a detached job.
8          ; 2==>Kill a detached job.
9          ;
10         002712 DETEMT:
11         ;
12         ; Jump to processing routine based on sub-function code.
13         ;
14         002712 113700 0000000 MOVBLK,RO ;GET SUB-FUNCTION CODE
15         002716 020027 0000003 CMP RO,#DETHI ;IS SUB-FUNCTION VALID?
16         002722 103003 BHIS 2$ ;BR IF NOT
17         002724 006300 ASL RO ;CONVERT TO WORD TABLE INDEX
18         002726 000170 003226' JMP @DETVEC(RO) ;ENTER PROCESSING ROUTINE
19         002732 005000 2$: CLR RO ;Return error code 0
20         002734 000137 0000000 JMP SETERR
21         ;
22         ; Start a new detached job.
23         ;
24         002740 032737 0000000 0000000 DET00: BIT #P0$DET,PRIVCO ;Is this user allowed to access detached jobs?
25         002746 001003 BNE 10$ ;Br if yes
26         002750 005000 CLR RO ;Return error code 0 if not
27         002752 000137 0000000 JMP SETERR
28         ;
29         ; Search for a free detached job line.
30         ;
31         002756 012701 0000000 10$: MOV #FSTD, R1 ;GET # OF FIRST DETACHED JOB LINE
32         002762 020127 0000000 3$: CMP R1,#LSTD ;HAVE WE CHECKED ALL?
33         002766 101043 BHI DETONE ;BR IF YES -- NONE FREE
34         002770 032761 0000000 0000000 BIT ##DILUP,LSW(R1) ;IS THIS LINE FREE?
35         002776 001403 BEQ 2$ ;BR IF YES
36         003000 062701 0000002 ADD #2,R1 ;GO TRY NEXT
37         003004 000766 BR 3$
38         ; Found a free detached job line.
39         ; Set name of start-up command file.
40         003006 013703 0000002 2$: MOV EMTBLK+2,R3 ;GET ADDRESS OF START-UP FILE NAME ARGUMENT
41         003012 010300 MOV R3,RO ;VALIDATE THE ADDRESS
42         003014 004737 0000000 CALL VALADB
43         003020 016102 0000000 MOV LSUCF(R1),R2 ;GET ADDRESS WHERE WE SHOULD STORE NAME
44         003024 012705 0000000 MOV #DETCBS,R5 ;GET COUNT OF MAX # CHARS TO MOVE
45         003030 004737 0000000 4$: CALL GETUCH ;GET CHAR FROM USER'S BUFFER
46         003034 110022 MOVBLK RO,(R2)+ ;STORE AS START-UP FILE NAME FOR LINE
47         003036 001402 BEQ 6$ ;BR IF MOVED NULL AT END OF NAME
48         003040 077505 SOB R5,4$ ;LOOP TO MOVE REST OF NAME
49         003042 105012 CLRBLK (R2) ;PUT NULL AT END OF NAME
50         ; Start up the line.
51         003044 005000 6$: CLR RO ;No secondary start-up command file
52         003046 OCALL INITLN ;INITIALIZE THE LINE
53         003054 103410 BCS DETONE ;BR IF INSUFFICIENT MEMORY TO START JOB
54         ; Set number of parent job
55         003056 113761 0000000 0000000 MOVBLK CORUSR,LPARNT(R1);Set # of parent job
56         ; Return detached job line number to user in R0.
57         003064 006201 ASR R1 ;CONVERT TO LINE NUMBER

```

```

58 003066 010137 000000G      MOV      R1,URO      ;RETURN IN USER'S RO
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 000002G      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 000002G      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
    
```

```

1          .SBTTL Shared Run-time control emt's
2          ;-----
3          ; All of the shared run-time control emt's have an emt function
4          ; code of 143. The sub-function code is stored in byte 0 of the
5          ; emt argument block.
6          ;
7          ; Sub-functions:
8          ; 0 ==> Associate or disassociate a run-time system.
9          ; 1 ==> Map part of a run-time system into job's virtual memory area.
10         ; 2 ==> Define shared run-time region for fast mapping.
11         ; 3 ==> Disable fast run-time region mapping (restore TRAP instr.)
12         ; 4 ==> Enable I/D space split
13         ; 5 ==> Disable I/D space split (return to all I-space)
14         ; 6 ==> Define PLAS region for fast mapping
15         ;
16 003234 113700 0000000 SSEMT:  MOVB   EMTBLK,RO      ;Get sub-function code
17 003240 020027 0000007          CMP    RO,#SSEHI    ;Valid sub-function code?
18 003244 103402          BLO   1$          ;Br if OK
19 003246 000137 0000000          JMP   BADEMT     ;Invalid sub-function
20 003252 006300          1$:  ASL    RO          ;Convert sub-function to word index
21 003254 000170 003260'          JMP   @SSEVEC(RO) ;Jump to appropriate routine
22         ;
23         ; Define jump vector table for mapping emts
24         ;
25 003260 003276' SSEVEC:  .WORD  SSASOC      ;0 - associate a run-time system
26 003262 003450'          .WORD  SSMAP       ;1 - map to shared run-time system
27 003264 003574'          .WORD  SSREGN     ;2 - define fast map to shared run-time
28 003266 003774'          .WORD  SSNOMP     ;3 - disable fast mapping (restore TRAP use)
29 003270 004132'          .WORD  ESPLIT    ;4 - enable I/D space split
30 003272 004172'          .WORD  DSPLIT    ;5 - restore all I space
31 003274 004220'          .WORD  SSPLAS     ;6 - define fast map to PLAS region
32         ;
33         SSEHI = <.-SSEVEC>/2 ;This and higher subfunctions are invalid
34         ;

```

```

1          .SBTTL Associate shared run-time system with job
2          ;-----
3          ; This emt is used to associate a shared run-time system with a job.
4          ; The form of the emt argument block is:
5          ;
6          ; .BYTE 0,143
7          ; .WORD name-pointer
8          ;
9          ; Where name-pointer is the address of a two word cell containing the
10         ; rad50 name of the run-time system to be associated.
11         ;
12 003276 013700 0000020 SSASOC: MOV EMTBLK+2,R0 ;GET ADDRESS OF NAME CELL
13 003302 001026         BNE 2$ ;BR IF WE ARE ASSOCIATING A RUN-TIME
14         ;
15         ; Name pointer is zero -- Disassociate all run-time systems
16         ;
17 003304 005037 0000000 CLR CURRDB ;NO ASSOCIATED RUN-TIME
18 003310 005002         CLR R2 ;INIT PAR INDEX
19 003312 005062 0000000 1$: CLR RPAR(R2) ;CLEAR PAR VALUE
20 003316 005062 0000000 CLR RPDR(R2) ;CLEAR PDR VALUE
21 003322 062702 0000002 ADD #2,R2 ;ADVANCE TO NEXT SET
22 003326 020227 0000016 CMP R2,#2*7 ;DONE ALL?
23 003332 101767         BLOS 1$ ;BR IF MORE TO CLEAR
24 003334 005737 0000000 TST VPLAS ;Do we have PLAS support?
25 003340 001403         BEQ 7$ ;Br if not
26 003342         OCALL MAPPLS ;Set up info about any PLAS regions
27 003350 004737 0000000 7$: CALL SETMAP ;RELOAD MAP REGISTERS FOR JOB
28 003354 000137 0000000 JMP EMTXIT ;FINISHED
29         ;
30         ; Associate new shared run-time system with job
31         ;
32 003360 004737 0000000 2$: CALL VALADW ;VALIDATE THE ADDRESS
33 003364 106520         MFPD (R0)+ ;GET 1ST WORD OF RUN-TIME NAME
34 003366 012604         MOV (SP)+,R4
35 003370 106520         MFPD (R0)+ ;GET 2ND WORD OF RUN-TIME NAME
36 003372 012605         MOV (SP)+,R5
37         ;
38         ; Search for run-time system descriptor with matching name
39         ;
40 003374 012703 0000000 MOV #RDB,R3 ;POINT TO 1ST RUN-TIME DESCRIPTOR BLOCK
41 003400 020327 0000000 3$: CMP R3,#RDBEND ;CHECKED ALL RUN-TIME DESCRIPTOR BLOCKS?
42 003404 103015         BHIS 6$ ;BR IF YES
43 003406 020463 0000000 CMP R4,RT$NAM(R3) ;COMPARE 1ST WORD OF NAME
44 003412 001003         BNE 5$ ;BR IF MISMATCH
45 003414 020563 0000020 CMP R5,RT$NAM+2(R3) ;COMPARE 2ND WORD OF NAME
46 003420 001403         BEQ 4$ ;BR IF FOUND RIGHT BLOCK
47 003422 062703 0000000 5$: ADD #RT$$SZ,R3 ;POINT TO NEXT RUN-TIME DESCRIPTOR BLOCK
48 003426 000764         BR 3$ ;GO CHECK NEXT ONE
49         ;
50         ; Found run-time descriptor block
51         ; Associate the run-time system with the job
52         ;
53 003430 010337 0000000 4$: MOV R3,CURRDB ;SAY RUN-TIME SYSTEM IS ASSOCIATED WITH JOB
54 003434 000137 0000000 JMP EMTXIT ;FINISHED
55         ;
56         ; Error -- Cannot find named run-time system
57         ; Return error code of 1

```

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

```

1          .SBTTL  Map shared run-time system into user's virtual address
2          ;-----
3          ; This emt is used to map a portion of a shared run-time system into
4          ; the virtual address region of the job.
5          ; The form of the emt is:
6          ;
7          ;       .BYTE  1,143
8          ;       .WORD  user-par          ;0 to 7
9          ;       .WORD  run-time-offset ;64-byte units
10         ;       .WORD  size             ;64-byte units
11         ;
12 003450  SSMAP:
13         ;
14         ; Call routine to check arguments and bias base address
15         ;
16 003450  004737  004004'  CALL  SSSTUP          ;Do common mapping setup
17         ;
18         ; Set up mapping information for each par
19         ;
20 003454  010500  3$:  MOV  R5,R0          ;CALCULATE # 64-BYTE BLOCKS
21 003456  160400          SUB  R4,R0          ; LEFT TO ALLOCATE
22 003460  003443          BLE  6$          ;BR IF FINISHED
23 003462  010462  000000G  MOV  R4,RPAR(R2)   ;SET PAR BASE OFFSET VALUE
24 003466  010462  000000G  MOV  R4,CUPAR0(R2) ;SAVE INFORMATION IN JOB CONTEXT BLOCK
25 003472  010462  000000G  MOV  R4,UPAR0(R2)  ;SET HARDWARE REGISTER
26 003476  020027  000200  CMP  R0,#200       ;EACH PAR CAN MAP UP TO 200 64-BYTE BLOCKS
27 003502  101402          BLOS 4$          ;BR IF WE CAN MAP ALL THAT'S LEFT
28 003504  012700  000200  MOV  #200,R0       ;ONLY MAP 200 THROUGH THIS PAR
29 003510  060004          4$:  ADD  R0,R4          ;ADVANCE BASE OFFSET NUMBER
30 003512  005300          DEC  R0            ;PDR VALUE IS ACTUAL-1
31 003514  000300          SWAB R0            ;POSITION LENGTH FOR PDR REGISTER
32 003516  052700  000002  BIS  #2,R0         ;SET READ-ALLOWED FLAG FOR PDR
33 003522  132763  000000G 000000G  BITB #RF$WRT,RT$FLG(R3); IS WRITE ACCESS TO BE ALLOWED?
34 003530  001402          BEQ  5$          ;BR IF NOT
35 003532  052700  000004  BIS  #4,R0         ;SET WRITE-ALLOWED FLAG FOR PDR
36 003536  042700  100261  5$:  BIC  #100261,R0   ;MAKE SURE UNUSED PDR BITS ARE ZERO
37 003542  010062  000000G  MOV  R0,RPDR(R2)  ;THIS IS PDR VALUE FOR PAR RANGE
38 003546  010062  000000G  MOV  R0,CUPDR0(R2);SAVE INFORMATION IN JOB CONTEXT BLOCK
39 003552  010062  000000G  MOV  R0,UPDR0(R2) ;SET HARDWARE MAPPING
40 003556  062702  000002  ADD  #2,R2         ;ADVANCE PAR INDEX #
41 003562  020227  000016  CMP  R2,#2*7      ;ONLY DO PAR UP TO # 7
42 003566  101732          BLOS 3$          ;BR IF MORE TO DO
43         ;
44         ; Finished
45         ;
46 003570  000137  000000G  6$:  JMP  EMTXIT      ;FINISHED

```


Define a run-time region for fast mapping

```

1          .SBTTL Define a run-time region for fast mapping
2          ;-----
3          ; Define a run-time region so that it can be quickly mapped using
4          ; the TRAP instruction.
5          ;
6          ; EMT argument list:
7          ; .BYTE 2,143
8          ; .WORD user-par ;PAR to map to region (0 to 7)
9          ; .WORD run-time-offset ;Offset within run-time (64-byte units)
10         ; .WORD size ;# 64-byte units to map
11         ; .WORD region-index ;0 to MAXSRD-1
12         ;
13 003574 SSREGN:
14         ;
15         ; Call routine to check arguments and bias base address
16         ;
17 003574 004737 004004' CALL SSSTUP ;Do common mapping setup
18         ;
19         ; At this point the following registers are set up:
20         ; R2 = PAR index number (2 * PAR).
21         ; R3 = Pointer to current run-time descriptor block.
22         ; R4 = Starting 64-byte physical address of region.
23         ; R5 = Ending 64-byte physical address of region.
24         ;
25         ; Make sure the region-index value is valid.
26         ;
27 003600 160405 SUB R4,R5 ;R5 now has # blocks to be mapped
28 003602 013700 000010G MOV EMTBLK+10,R0 ;Get specified region #
29 003606 020027 000000G 1$: CMP R0,#MAXSRD ;Is the number valid?
30 003612 103404 BLO 11$ ;Br if ok
31 003614 012700 000002 MOV #EC2,R0 ;Return error code 2
32 003620 000137 000000G JMP SETERR
33         ;
34         ; Remember which PAR is set by this region
35         ;
36 003624 110260 000000G 11$: MOVB R2,SR$PX(R0) ;Save PAR index (2*PAR #)
37 003630 105060 000000G CLRB SR$FLG(R0) ;No special mapping flags yet
38         ;
39         ; Set the PAR value in the region descriptor
40         ;
41 003634 006300 ASL R0 ;Get word table index
42 003636 005060 000000G CLR SR$WCB(R0) ;This is a shared run-time region (not PLAS)
43 003642 010460 000000G MOV R4,SR$PAR(R0) ;Set base PAR value for region
44         ;
45         ; Set the PDR value for the region
46         ;
47 003646 010501 MOV R5,R1 ;Get # blocks remaining to map
48 003650 020127 000200 CMP R1,#200 ;Will it all fit thru 1 par?
49 003654 101402 BLOS 2$ ;Br if ok
50 003656 012701 000200 MOV #200,R1 ;If not, decrease to 1 par worth
51 003662 160105 2$: SUB R1,R5 ;This many left to map next time
52 003664 060104 ADD R1,R4 ;Bump up base address for next time
53 003666 005301 DEC R1 ;PDR size is 1 less than wanted
54 003670 000301 SWAB R1 ;Position size for PDR value
55 003672 132763 000000G 000000G BITB #RF$WRT,RT$FLG(R3);Can we write to shared run-time?
56 003700 001402 BEQ 3$ ;Br if not
57 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 BISH     #SR.MDR,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     SSNDMP: CLRB     DOTRMP ;Say mapping not in effect
89 004000 000137 000000G     JMP      EMTXIT

```

Define a run-time region for fast mapping

```

1          ; -----
2          ; Common routine used to process arguments for shared run-time
3          ; mapping EMT's.
4          ;
5          ; Outputs:
6          ;   R2 = PAR index number (2 * PAR).
7          ;   R3 = Pointer to current run-time descriptor block.
8          ;   R4 = Starting 64-byte physical address of region.
9          ;   R5 = Ending 64-byte physical address of region.
10         ;   URO = Size of area that is being mapped.
11         ;
12 004004  SSSTUP:
13         ;
14         ; Check the specified PAR number
15         ;
16 004004 013702 0000020  MOV     EMTBLK+2,R2    ;Get PAR #
17 004010 020227 0000007  CMP     R2,#7        ;Must be in range 0 to 7
18 004014 101404          BLOS   7#           ;Br if ok
19 004016 012700 177767  MOV     #-11,R0      ;Fatal error if other value
20 004022 000137 0000000  JMP     SETERR
21 004026 006302          7$:    ASL     R2        ;Convert par # to word table index
22         ;
23         ; Get pointer to descriptor for current shared run-time system
24         ;
25 004030 013703 0000000  MOV     CURRDB,R3    ;Get address of associated run-time descriptor
26 004034 001004          BNE    1#           ;Better be one
27 004036 012700 0000001  MOV     #EC1,R0      ;If not then return error code 1
28 004042 000137 0000000  JMP     SETERR
29         ;
30         ; Get physical memory address
31         ;
32 004046 013704 0000040  1$:    MOV     EMTBLK+4,R4 ;Get run-time base offset
33 004052 020463 0000000  CMP     R4,RT$TOP(R3) ;Make sure it is within run-time area
34 004056 103404          BLO    3#           ;Br if ok
35 004060 012700 0000005  MOV     #EC5,R0      ;Return error 5 if too high
36 004064 000137 0000000  JMP     SETERR
37 004070 013705 0000060  3$:    MOV     EMTBLK+6,R5 ;Get size of region to be mapped
38 004074 060405          ADD     R4,R5        ;Get offset above top of region to be mapped
39 004076 066304 0000000  ADD     RT$BAS(R3),R4 ;Bias the offsets
40 004102 066305 0000000  ADD     RT$BAS(R3),R5
41         ;
42         ; Get amt of memory to be mapped
43         ;
44 004106 020563 0000000  CMP     R5,RT$TOP(R3) ;Is top of request above top of run-time?
45 004112 101402          BLOS   2#           ;Br if not
46 004114 016305 0000000  MOV     RT$TOP(R3),R5 ;Don't map beyond top of run-time
47         ;
48         ; Return actual mapped size to user in R0
49         ;
50 004120 010537 0000000  2$:    MOV     R5,URO   ;Return size of mapped region in user's R0
51 004124 160437 0000000  SUB     R4,URO
52         ;
53         ; Finished
54         ;
55 004130 000207          RETURN

```

```

1          .SBTTL  ESPLIT -- Enable user D-space
2          ;-----
3          ; Enable/Disable I and D space split
4          ; When one or more of a job's PARs are mapped away to a shared run-time
5          ; the root memory is not released, but is ordinarily inaccessible. If
6          ; I/D space split is enabled, then the root memory which was mapped
7          ; away by a shared run-time can be accessed as D-space. This is only valid
8          ; on machines which support I and D space (have memory management register
9          ; 3 -- SR3FLG not 0 and have D-space APR's -- IDSFLG not 0).
10         ;
11         ; Enabling D-space is done by setting the LSW11 bit which says the user
12         ; wants to split, then calling SUTOP with the current size, which then
13         ; calls SETMAP and copies the ordinary I-space APRs into the D-space APRs
14         ; and enables user D-space.
15         ;
16         ; The form of the EMT is:
17         ;     .BYTE  4,143
18         ;
19 004132 105737 000000G  ESPLIT: TSTB   IDSFLG      ; Does hardware support I and D space?
20 004136 001411          BEQ     SPLITR     ; If not, br to split error return
21 004140 052761 000000G 000000G  BIS     #$UDSPC,LSW11(R1) ; Set flag to enable D-space mapping
22 004146 013700 000000G  MOV     ODTBAS,RO      ; Point to word beyond normal address limit
23 004152 004737 000000G  CALL   SUTOP          ; This will do the real work
24 004156 000137 000000G  JMP    EMTXIT
25         ;
26         ; Error, hardware does not support I and D space
27         ;
28 004162 012700 000000G  SPLITR: MOV    #EC6,RO      ; Signal error code 1
29 004166 000137 000000G  JMP    SETERR
30         ;
31         ;-----
32         ; Disable D-space split, return all mapping to I-space
33         ; This just requires turning off the hardware D-space enable bit,
34         ; the D-space APRs will then be ignored so they don't require cleanup.
35         ;
36         ; The form of the EMT is:
37         ;     .BYTE  5,143
38         ;
39 004172 105737 000000G  DSPLIT: TSTB   IDSFLG      ; Does hardware support I and D space?
40 004176 001771          BEQ     SPLITR     ; Br if not
41 004200 042761 000000G 000000G  BIC     #$UDSPC,LSW11(R1) ; Clear user D-space in use flag
42 004206 042737 000000G 000000G  BIC     #USDSPC,@#SR3MMR ; And disable it in hardware
43 004214 000137 000000G  JMP    EMTXIT

```

Define a PLAS region for fast mapping

```

1          .SBTTL Define a PLAS region for fast mapping
2          ;-----
3          ; Define a PLAS window so it can be quickly mapped using the TRAP
4          ; instruction. The region is not limited to 8Kb. All checking related
5          ; to PAR numbers and mapped length is assumed to have been done
6          ; when creating the window.
7          ;
8          ; EMT argument list:
9          ; .BYTE 6,143
10         ; .BYTE W.NID,fmrnum ;PLAS window ID, <0 to MAXSRD-1>
11         ; .WORD W.NRID ;R.GID to which window is associated
12         ;
13 004220 113701 0000020 SSPLAS: MOVB EMTBLK+2,R1 ;Get window ID
14 004224 005301 DEC R1 ;Make it 0 indexed
15 004226 020127 0000000 CMP R1,#NUMWCB ;Is window ID in valid range? (1-NUMWCB)
16 004232 103403 BLO 2$ ;Br if OK
17 004234 012700 0000003 1$: MOV #EC3,R0 ;Invalid window ID
18 004240 000421 BR 7$ ;Return with error
19         ;
20         ; Verify that fast map region number is valid
21         ;
22 004242 113700 0000030 2$: MOVB EMTBLK+3,R0 ;Get fast map region number
23 004246 020027 0000000 CMP R0,#MAXSRD ;Is number in valid range?
24 004252 103403 BLO 3$ ;Br if OK
25 004254 012700 0000002 21$: MOV #EC2,R0 ;Invalid fast map region #
26 004260 000411 BR 7$ ;Return with error
27         ;
28         ; Convert window ID into ptr to window control block
29         ;
30 004262 070127 0000000 3$: MUL #WC#$SZ,R1 ;And offset into window control blocks
31 004266 062701 0000000 ADD #WCBBAS,R1 ;Point to specified window control block
32 004272 016105 0000000 MOV WC$SIZ(R1),R5 ;Get # 64-byte blocks to be mapped
33 004276 001004 BNE 4$ ;Br if window is allocated
34 004300 012700 0000001 31$: MOV #EC1,R0 ;Else say window is undefined
35 004304 000137 0000000 7$: JMP SETERR ;Return with error
36         ;
37         ; Now move window control information into fast map cells
38         ;
39 004310 013702 0000040 4$: MOV EMTBLK+4,R2 ;Get ptr to region control block
40 004314 004737 004576' CALL CHKRCB ;Verify that it is valid
41 004320 103767 BCS 31$ ;Br if invalid
42 004322 032762 0000000 0000000 BIT #RC$LCG,RC$FLG(R2) ;Is this region swappable?
43 004330 001003 BNE 41$ ;Br if not
44 004332 012700 0000004 MOV #EC4,R0 ;Can't fast map swappable regions!
45 004336 000762 BR 7$ ;Go do error
46 004340 005046 41$: CLR -(SP) ;Assume I-space mapping
47 004342 032762 0000000 0000000 BIT #RC$DSP,RC$FLG(R2) ;Should we use D-space instead?
48 004350 001402 BEQ 44$ ;Br if not
49 004352 112716 0000000 MOVB #SR.DSP,(SP) ;If using D-space, init flag byte
50         ;
51         ; Ensure that window won't exceed region bounds and update WCB cells
52         ;
53 004356 016204 0000000 44$: MOV RC$LEN(R2),R4 ;Get size of region
54 004362 166104 0000000 SUB WC$OFF(R1),R4 ;Figure maximum allowed window length
55 004366 020504 CMP R5,R4 ;Would request exceed region?
56 004370 103401 BLO 46$ ;Br if not
57 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      MOVVB   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$OFF(R1),R4  ;Add offset to PLAS window
69 004432 116102 000000G      MOVVB   WC$PAR(R1),R2  ;Get PAR index # (2*PAR #)
70 ; At this point, R0 has the trap #, NOT the trap index
71 004436 110061 000000G      MOVVB   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$:  MOVVB   R2,SR$PX(R0)  ;Set PAR index into fast map cell
77 004460 111660 000000G      MOVVB   (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.MDR,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  MOVVB   #1,DOTRMP      ;Enable fast mapping
104 004572 000137 000000G      JMP     EMTXIT

```

CHKRCB -- Check address of Region Control Block

```

1                                     .SBTTL  CHKRCB -- Check address of Region Control Block
2                                     ;-----
3                                     ;  CHKRCB is called to verify that the address of a region control
4                                     ;  block is valid.  This routine is a similar to that in TSPLAS.
5                                     ;
6                                     ;  Inputs:
7                                     ;  R2 = Region control block address to be checked
8                                     ;
9 004576 032702 000001  CHKRCB: BIT      #1,R2          ;Make sure address is even
10 004602 001014          BNE      1$          ;Br if odd
11 004604 020227 000000G  CMP      R2,#RCBBAS      ;Make sure it is within proper region
12 004610 103403          BLO      2$
13 004612 020227 000000G  CMP      R2,#RCBEND
14 004616 103410          BLO      3$
15 004620 020237 000000G  2$:  CMP      R2,SHRRCB      ;See if this is a shared RCB
16 004624 103403          BLO      1$
17 004626 020237 000000G  CMP      R2,SHRRCN
18 004632 101402          BLOS     3$
19                                     ;
20                                     ;  Address is invalid
21                                     ;
22 004634 000261          1$:  SEC
23 004636 000401          BR      9$
24                                     ;
25                                     ;  Address is ok
26                                     ;
27 004640 000241          3$:  CLC
28                                     ;
29 004642 000207          9$:  RETURN
30

```

```

1          .SBTTL  Spooling control
2          ;-----
3          ; This EMT is used to control the spooling operation.
4          ;
5          ; Form of the EMT argument block:
6          ;   .BYTE   chan,151
7          ;   .WORD   sub-function
8          ;   .WORD   value
9          ;
10         ; Where "chan" is an open channel whose file is to be affected,
11         ; and "sub-function" indicates which function is to be performed.
12         ;
13         004644 013703 000000G SPLEMT: MOV     CHNADR,R3      ;Get address of specified channel
14         004650 032763 000000G 000000G BIT     #CS$OPN,C.CSW(R3);Is the channel open?
15         004656 001004          BNE     1$             ;Br if yes
16         004660 012700 000003  MOV     #EC3,R0      ;Channel not open
17         004664 000137 000000G      JMP     SETERR
18         ;
19         ; Ignore this EMT if the channel is not opened to a spooled device
20         ;
21         004670 032763 000000G 000000G 1$: BIT     #CS$SPL,C.CSW(R3);Is channel opened to a spooled device?
22         004676 001412          BEQ     9$             ;Ignore the EMT if not
23         ;
24         ; Get sub-function code and branch off to processing routine
25         ;
26         004700 013700 000002G      MOV     EMTBLK+2,R0   ;Get EMT sub-function code
27         004704 006300          ASL     R0            ;Convert to word-table index
28         004706 020027 000006  CMP     R0,#MXSFX    ;Is index ok?
29         004712 103402          BLO    2$           ;Br if ok
30         004714 000137 000000G      JMP     BADEMT      ;Invalid sub-function code
31         004720 000170 005020' 2$: JMP     @SPLFNV(R0) ;Enter processing routine
32         004724 000137 000000G 9$: JMP     EMTXIT
33         ;
34         ; Set Hold/Nohold flag for spool file
35         ;
36         004730 105737 000000G SPSTHL: TSTB   NSPLDV   ;Are there any spooled devices?
37         004734 001405          BEQ    9$           ;Br if not
38         004736 013702 000004G      MOV     EMTBLK+4,R2 ;Get [no]hold flag
39         004742          DCALL  SPLSHF   ;Set hold flag for file
40         004750 000137 000000G 9$: JMP     EMTXIT   ;Finished
41         ;
42         ; Set Hold/Nohold flag for spool file
43         ;
44         004754 105737 000000G SPSTFL: TSTB   NSPLDV   ;Are there any spooled devices?
45         004760 001404          BEQ    9$           ;Br if not
46         004762 013702 000004G      MOV     EMTBLK+4,R2 ;Get [no]flagpage flag
47         004766 000137 000000G      JMP     SPLSFF     ;Set flagpage flag for device
48         004772 000137 000000G 9$: JMP     EMTXIT   ;Finished
49         ;
50         ; Set Hold/Nohold flag for spool file
51         ;
52         004776 105737 000000G SPSTWD: TSTB   NSPLDV   ;Are there any spooled devices?
53         005002 001404          BEQ    9$           ;Br if not
54         005004 013702 000004G      MOV     EMTBLK+4,R2 ;Get wide/narrow flag
55         005010 000137 000000G      JMP     SPLSWF     ;Set wide/narrow flag for device
56         005014 000137 000000G 9$: JMP     EMTXIT   ;Finished
57         ;

```



```
58          ; Vector of routine addresses for sub-function codes
59          ;
60 005020 004730' SPLFNV: .WORD  SPSTHL          ; 0 -- Set [no]hold flag for file
61 005022 004754'          .WORD  SPSTFL          ; 1 -- Turn on flag pages for device
62 005024 004776'          .WORD  SPSTWD          ; 2 -- Set flag page wide/narrow for device
63          000006          MXSFX  =          .-SPLFNV          ;# functions
```

```

1          .SBTTL  CL line control EMT's
2          ;-----
3          ;
4          ; Assorted CL EMT's -- some of which duplicate CL spfun services, but
5          ; are also implemented as EMT's to avoid the requirement to have a
6          ; channel open to the CL unit.
7          ;
8          ; The form of the argument block is:
9          ;
10         ; .BYTE    n,155
11         ; .WORD    CL-unit-number          ;0 to (CLTOTL-1)
12         ; .WORD    time-sharing-line-number ;Only used for CLASN
13         ;
14 005026  CLEMT:
15         ;
16         ; Require TERMINAL privilege to manipulate CL units
17         ;
18 005026  032737  000000G 000000G      BIT    #P2$TRM,PRIVC2 ;Is this user privileged?
19 005034  001003                BNE    12$          ;Br if yes
20 005036  012700  000001          MOV    #EC1,R0      ;Error 1 if user not privileged
21 005042  000436                BR     30$
22         ;
23         ; Check to see if the specified CL unit number is valid
24         ;
25 005044  013702  000002G          12$:   MOV    EMTBLK+2,R2    ;Get specified CL unit number
26 005050  020227  000000G          CMP    R2,#CLTOTL  ;Is this in the proper range?
27 005054  103403                BLO   1$          ;Br if ok
28 005056  012700  000002          MOV    #EC2,R0      ;Error 2
29 005062  000426                BR     30$
30 005064  006302          1$:   ASL    R2          ;Convert CL unit # to table index
31         ;
32         ; Now jump off to the individual processing routines
33         ;
34 005066  113700  000000G          MOVB   EMTBLK,R0    ;Get subfunction code
35 005072  020027  000003          CMP    R0,#CLEMMX  ;Is it valid?
36 005076  103402                BLO   2$          ;Br if OK
37 005100  005000                CLR    R0          ;Else return ECO
38 005102  000416                BR     30$
39         ;
40 005104  006300          2$:   ASL    R0          ;Convert subfunction code to index
41 005106  001416                BEQ   3$          ;Br if we don't need to check for hdw or priv
42 005110  032737  000000G 000000G      BIT    #P0$BYP,PRIVC0 ;Does issuer have BYPASS privilege?
43 005116  001003                BNE   21$         ;Br if yes
44 005120  012700  000010          MOV    #EC8,R0      ;Oh, too bad. Must have BYPASS
45 005124  000405                BR     30$
46 005126  016201  000000G          21$:  MOV    CL$LIX(R2),R1 ;Get index of T-S line CL is connected to
47 005132  001004                BNE   3$          ;Br if it is connected to a line
48 005134  012700  000007          MOV    #EC7,R0      ;Oops, not connected to hardware!
49 005140  000137  000000G          30$:  JMP    SETERR
50         ;
51 005144  000170  005150'          3$:   JMP    @CLCODE(R0) ;Enter the processing routine
52         ;
53 005150  005154'          CLCODE: .WORD   CLASN    ;0 Assign a CL unit to a line
54 005152  005674'          .WORD   CLCLR    ;1 Clear XOFF status (=== SPFUN 201)
55 005154  005710'          .WORD   CLRES    ;2 Reset CL unit (=== SPFUN 265)
56         CLEMMX = <.-CLCODE>/2

```

CLASN -- Assign a CL unit to a line

```

1          .SBTTL          CLASN -- Assign a CL unit to a line
2          ;-----;
3          ; Assign a CL unit to a line.
4          ; The form of the EMT is:
5          ;
6          ;       .BYTE    0,155
7          ;       .WORD    CL_unit_number
8          ;       .WORD    line_number
9          ;
10         CLASN:
11         ;
12         ; Check to see if the specified line number is ok
13         ;
14         005156 013704 000004G      MOV      EMTBLK+4,R4      ;Get specified line number
15         005162 001451              BEQ      5$              ;LINE=0 ==> Free this unit
16         005164 006304              ASL      R4              ;Convert line # to line table index
17         005166 020427 000000G      CMP      R4,#LSTHL      ;Must be a valid hardware line
18         005172 101042              BHI      2$              ;Br if invalid
19         005174 032764 000000G 000000G BIT      ##HARD,LSW3(R4) ;Is this line connected to hardware?
20         005202 001436              BEQ      2$              ;Br if not
21         005204 020264 000000G      CMP      R2,LCLUNT(R4)  ;Is this CL unit already assigned to this line
22         005210 001535              BEQ      22$             ;Br if yes -- nothing else needs to be done
23         005212 032764 000000G 000000G BIT      ##DEAD,LSW3(R4) ;Is this line installed?
24         005220 001027              BNE      2$              ;Br if not
25         005222 005764 000000G      TST      LCLUNT(R4)    ;Is this line in use by another CL unit?
26         005226 002404              BLT      4$              ;Br if not
27         005230 012700 000004      MOV      #EC4,R0        ;Error 4 if line already being used by CL
28         005234 000137 000000G      30$:    JMP      SETERR
29         005240 020427 000000G      4$:    CMP      R4,#LSTPL      ;Connecting to a primary time-sharing line?
30         005244 101007              BHI      3$              ;Br if not
31         005246 032764 000000G 000000G BIT      ##DILUP,LSW(R4) ;Is this line in use by time-sharing user?
32         005254 001414              BEQ      5$              ;Br if not
33         005256 012700 000005      MOV      #EC5,R0        ;Error 5 if line is active as a time-sharing
34         005262 000764              BR      30$
35         005264 020427 000000G      3$:    CMP      R4,#FSTIOL      ;Are we assigning to a reserved CL line?
36         005270 103403              BLO      2$              ;Br if not
37         005272 020427 000000G      CMP      R4,#LSTIOL      ;Is this a valid CL line?
38         005276 101403              BLOS     5$              ;Br if yes
39         005300 012700 000003      2$:    MOV      #EC3,R0        ;Error 3 if invalid line number
40         005304 000753              BR      30$
41         ;
42         ; The line number is ok.
43         ; Deassign the CL unit from its current line association.
44         ;
45         005306 016203 000000G      5$:    MOV      CL$LIX(R2),R3 ;Get line # currently assoc with this CL unit
46         005312 001471              BEQ      6$              ;Br if line is not currently assigned
47         005314 005762 000000G      TST      CL$XLN(R2)     ;Is CL unit in use for cross connect?
48         005320 001005              BNE      23$             ;Br if yes
49         005322 016200 000000G      MOV      CL$RQH(R2),R0  ;Anything in read queue head
50         005326 056200 000000G      BIS      CL$WQH(R2),R0  ;or write queue head for this CL unit?
51         005332 001403              BEQ      11$             ;Br if not
52         005334 012700 000006      23$:   MOV      #EC6,R0        ;Error 6 if CL unit is currently busy
53         005340 000735              BR      30$
54         005342 004737 006026'      11$:   CALL     CLWAIT          ;Wait for output characters to be sent
55         005346 010205              MOV      R2,R5          ;Get CL unit index to R5
56         005350              DCALL   CLREST          ;Reset CL unit status
57         005356 042762 000000G 000000G BIC      #CD#DTR,CL$(OPT(R2) ;Request DTR down

```

CLASN -- Assign a CL unit to a line

```

58 005364          DCALL  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      MOV    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    ##BBIT,R3   ;Does this device want 8-bit support?
101 005606 001402          BEQ    10$         ;Br if not
102 005610 052700 000000G      BIS    #CO$BBT,R0   ;Enable 8 bit support
103 005614 010062 000000G      10$:  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 000000G   CLR   CL$EPP(R2)
117                          ;
118                          ; Finished
119                          ;
120 005670 000137 000000G   20$:  JMP   EMTXIT      ;Finished assigning the CL unit
```

CLCLR -- Clear XOFF sent and send XON

```

1          .SBTTL          CLCLR -- Clear XOFF sent and send XON
2          ;-----
3          ; This routine is the EMT equivalent of CL SPFUN 201
4          ;
5          ; Emt argument block:
6          ;   .BYTE  1,155
7          ;   .WORD  CL-unit-number
8          ;
9          ; Inputs:
10         ;   R1      TS line in use as CL unit
11         ;   R2      CL unit index
12         ;
13 005674  CLCLR:
14 005674  010205      MOV     R2,R5          ;Need to get CL unit index into R5
15 005676      OCALL   CLCLR          ;Call TSCLD routine to clear CL unit
16 005704  000137  000000G  JMP     EMTXIT
17         ;
18         .SBTTL          CLRES -- Reset CL line
19         ;-----
20         ; This routine is the EMT equivalent of CL SPFUN 265
21         ;
22         ; EMT argument block:
23         ;   .BYTE  2,155
24         ;   .WORD  CL-unit-number
25         ;
26         ; Inputs:
27         ;   R1      TS line in use as CL unit
28         ;   R2      CL unit index
29 005710  CLRES:
30 005710  010205      MOV     R2,R5          ;Need to get CL unit index into R5
31 005712      OCALL   CLREST          ;Call TSCLD routine to reset CL unit
32 005720  000137  000000G  JMP     EMTXIT

```

```

1          .SBTTL  CLCLOS -- Close channel opened to CL unit
2          ;-----
3          ; This routine is called to determine if the channel being closed is
4          ; assigned to a CL unit.  If it is, the job is suspended until all output
5          ; to the CL unit has been completed.
6          ;
7 005724 010246 CLCLOS: MOV      R2, -(SP)
8 005726 010546          MOV      R5, -(SP)
9          ;
10         ; Determine if this channel is opened to a CL unit
11         ;
12 005730 013702 000000G          MOV      CHNADR, R2          ;Get address of current channel block
13 005734 016200 000000G          MOV      C, CSW(R2), R0          ;Get channel status word
14 005740 032700 000000G          BIT      #CS$SPL, R0          ;Is this a spooled device?
15 005744 001025          BNE      9$          ;Br if yes -- Don't wait for output to finish
16 005746 116202 000000G          MOVB    C, DEVQ(R2), R2          ;Get unit number
17 005752 042702 177770          BIC      #^C<7>, R2          ;Clear all but unit number
18 005756 042700 000000C          BIC      #^C<CS$NMX>, R0          ;Mask out all but device table index
19 005762 020037 000000G          CMP      R0, CLDEVX          ;Is this a CL device?
20 005766 001405          BEQ      1$          ;Br if yes
21 005770 020037 000000G          CMP      R0, C1DEVX          ;Is this a C1 device?
22 005774 001011          BNE      9$          ;Br if not
23 005776 062702 000010          ADD      #8, R2          ;Bias unit number by 8 for C1
24         ;
25         ; This is a CL device being closed.
26         ; Wait for all output to be completed.
27         ;
28 006002 006302          1$: ASL      R2          ;Convert to word table index
29 006004 004737 006026'          CALL    CLWAIT          ;Wait for all output to finish
30 006010 010205          MOV      R2, R5          ;Get CL unit index to R5
31 006012          OCALL   CLREST          ;Reset CL unit status
32         ;
33         ; Finished
34         ;
35 006020 012605          9$: MOV      (SP)+, R5
36 006022 012602          MOV      (SP)+, R2
37 006024 000207          RETURN

```

CLWAIT -- Wait for CL output to finish

```

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 0000000 0000000 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 0000000      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 0000000 0000000      BIT      #CM$EFP, CL$STA(R2);Are we doing end-of-file processing?
24 006060 001004          BNE      2$          ;Br if yes
25 006062 032760 0000000 0000000      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 0000000 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

```



```

1
2
3
4
5
6
7
8
9
10
11
12
13
14 006120 113705 000000G
15 006124 120527 000003
16 006130 101402
17 006132 000137 000000G
18 006136 006305
19 006140 000175 006144'
20
21 006144 006154'
22 006146 006366'
23 006150 006402'
24 006152 006416'

```

```

.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.
;
PMEMT:  MOVB   EMTBLK,R5      ;GET SUB-FUNCTION CODE FOR EMT
        CMPB   R5,#3        ;VALID SUB-FUNCTION CODE?
        BLOS  1$           ;BR IF OK
        JMP    BADEMT       ;INVALID EMT CODE
1$:     ASL    R5            ;CONVERT TO WORD-TABLE INDEX
        JMP    @PMRVEC(R5)  ;ENTER PROCESSING ROUTINE
;
PMRVEC: .WORD  PMINIT       ;0 -- INITIALIZE
        .WORD  PMSTRT      ;1 -- START MONITORING
        .WORD  PMSTOP      ;2 -- SUSPEND MONITORING
        .WORD  PMEND       ;3 -- END ANALYSIS

```

```

1          ; -----
2          ; Initialize performance monitoring.
3          ;
4          ; Arguments:
5          ; 1. Base address of region being monitored.
6          ; 2. Top address of region being monitored.
7          ; 3. Number of bytes per cell in histogram.
8          ; 4. PF$ control flags.
9          ;
10         ; Errors:
11         ; 0 = Performance monitor already in use.
12         ; 1 = Performance monitor not generated in.
13         ;
14 006154 005737 000000G PMINIT: TST      PMCELS      ; WAS PERFORMANCE MONITOR GENNED IN?
15 006160 001004          BNE      5$          ; BR IF YES
16 006162 012700 000001  MOV      #EC1,R0      ; RETURN ERROR CODE 1
17 006166 000137 000000G  JMP      SETERR
18 006172 113702 000000G  5$:  MOVB   PMUSER,R2    ; IS ANYONE DOING PERFORMANCE MONITORING NOW?
19 006176 001411          BEQ      1$          ; BR IF NOT
20 006200 120237 000000G  CMPB   R2,CORUSR     ; ARE WE THE ONE?
21 006204 001406          BEQ      1$          ; BR IF YES
22 006206 005000          CLR      R0          ; PM IN USE BY SOMEONE ELSE -- RETURN ERROR 0
23 006210 006202          ASR      R2          ; RETURN # OF JOB THAT OWNS PM IN USER'S R0
24 006212 010237 000000G  MOV      R2,UR0
25 006216 000137 000000G  JMP      SETERR
26 006222 113737 000000G 000000G 1$:  MOVB   CORUSR,PMUSER ; SAY WE ARE USING PM FACILITY
27 006230 005037 000000G  CLR      PMRUN      ; DON'T START MONITORING YET
28         ;
29         ; Save information about region being monitored.
30         ;
31 006234 013705 000002G  MOV      EMTBLK+2,R5  ; GET BASE ADDRESS OF REGION
32 006240 010537 000000G  MOV      R5,PMBASE
33 006244 013703 000004G  MOV      EMTBLK+4,R3  ; GET TOP ADDRESS OF REGION
34 006250 010337 000000G  MOV      R3,PMTOP
35 006254 013737 000010G 000000G  MOV      EMTBLK+10,PMFLGS ; GET CONTROL FLAGS
36 006262 160503          SUB      R5,R3        ; GET # BYTES IN REGION BEING MONITORED
37 006264 000241          CLC
38 006266 006003          ROR      R3          ; CONVERT TO # WORDS
39 006270 005002          CLR      R2          ; SET FOR DIVIDE
40 006272 071237 000000G  DIV      PMCELS,R2    ; DIVIDE BY # CELLS IN PM DATA AREA
41 006276 005703          TST      R3          ; IS REMAINDER ZERO?
42 006300 001401          BEQ      2$          ; BR IF YES
43 006302 005202          INC      R2          ; ROUND UP QUOTIENT IF NOT
44 006304 006302          2$:  ASL      R2          ; GET # BYTES PER PM CELL
45 006306 020237 000006G  CMP      R2,EMTBLK+6 ; COMPARE WITH REQUESTED # BYTES PER CELL
46 006312 103002          BHS      4$          ; USE LARGER VALUE
47 006314 013702 000006G  MOV      EMTBLK+6,R2
48 006320 010237 000000G  4$:  MOV      R2,PMNBPC  ; NUMBER OF BYTES PER CELL
49         ;
50         ; Zero performance monitoring count cells.
51         ; (Enter system state so that we are running on interrupt stack and can
52         ; access PM data area through PAR6. )
53         ;
54 006324 012700 006362'  MOV      #6$,R0      ; GO TO 6$ ON EXIT FROM SYSTEM STATE
55 006330 004737 000000G  CALL     ENSYS        ; ENTER SYSTEM STATE
56 006334 000000G  .WORD   FP$MOV        ; Set fork processing priority
57 006336 013700 000000G  MOV      PMCELS,R0    ; GET # CELLS
    
```

```
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
```

```

1          ; -----
2          ; Start monitoring execution.
3          ;
4 006366 004737 006616' PMSTRT: CALL    PMCHKU    ; MAKE SURE WE ARE PM USER
5 006372 005237 000000G          INC      PMRUN      ; START MONITORING
6 006376 000137 000000G          JMP      EMTXIT
7          ; -----
8          ;
9          ; Suspend performance monitoring.
10         ;
11 006402 004737 006616' PMSTOP: CALL   PMCHKU    ; MAKE SURE WE ARE PM USER
12 006406 005037 000000G          CLR      PMRUN      ; SUSPEND PM MONITORING
13 006412 000137 000000G          JMP      EMTXIT
14         ; -----
15         ;
16         ; Return results of performance monitoring.
17         ;
18         ; Arguments:
19         ; 1. Address of vector to receive parameter values.
20         ; 2. Address of area to receive count vector.
21         ; 3. Size of area to receive count vector (# bytes).
22         ;
23         ; Values returned in parameter vector:
24         ; 1. Base address of monitored region.
25         ; 2. Top address of monitored region.
26         ; 3. Number of bytes per count cell.
27         ; 4. PF# control and status flags.
28         ;
29         ; Errors:
30         ; 0 = This job is not doing a performance monitor.
31         ; 1 = Area for count vector is too small.
32         ;
33 006416 004737 006616' PMEND:  CALL   PMCHKU    ; MAKE SURE WE ARE PM USER
34 006422 005037 000000G          CLR      PMRUN      ; STOP MONITORING
35 006426 005037 000000G          CLR      PMUSER     ; SAY PERFORMANCE MONITOR IS FREE
36         ;
37         ; Return parameter values
38         ;
39 006432 013701 000002G          MOV      EMTBLK+2,R1  ; GET ADDRESS OF USER'S RECEIVING AREA
40 006436 010100          MOV      R1,RO      ; VALIDATE THE ADDRESS
41 006440 004737 000000G          CALL   VALADW
42 006444 013746 000000G          MOV      PMBASE,-(SP) ; RETURN BASE ADDRESS
43 006450 106621          MTPD    (R1)+
44 006452 013746 000000G          MOV      PMTOP,-(SP)  ; RETURN TOP ADDRESS
45 006456 106621          MTPD    (R1)+
46 006460 013746 000000G          MOV      PMNBPC,-(SP) ; NUMBER OF BYTES PER CELL
47 006464 106621          MTPD    (R1)+
48 006466 013746 000000G          MOV      PMFLGS,-(SP) ; STATUS FLAGS
49 006472 106621          MTPD    (R1)+
50         ;
51         ; Return count values.
52         ;
53         ; Enter system state so that we are running on the interrupt stack and
54         ; can map to PM data area through PAR6.
55         ;
56 006474 012700 006576'          MOV      #3$,RO      ; GO TO 3$ ON RETURN FROM SYSTEM STATE
57 006500 004737 000000G          CALL   ENSYS        ; ENTER SYSTEM STATE

```

```

58 006504 000000G .WORD FP$MOV ;Fork processing priority
59 006506 013702 000004G MOV EMTBLK+4,R2 ;GET ADDRESS OF USER'S BUFFER
60 006512 010200 MOV R2,R0 ;VALIDATE THE ADDRESS
61 006514 004737 000000G CALL VALADW
62 006520 013703 000006G MOV EMTBLK+6,R3 ;GET # BYTES IN USER'S AREA
63 006524 006203 ASR R3 ;CONVERT TO # WORDS
64 006526 013704 000000G MOV PMBASE,R4 ;GET ADDRESS OF BASE OF MONITORED REGION
65 006532 012705 000000G MOV #VPAR6,R5 ;ACCESS PM DATA AREA THROUGH PAR6 REGION
66 006536 013737 000000G 000000G MOV PMPAR,@#KPAR6 ;SET PAR6 TO MAP TO PM DATA AREA
67 006544 005037 006614' CLR PMOVFR ;SAY USER BUFFER HAS NOT OVERFLOWED
68 006550 012546 1$: MOV (R5)+,-(SP) ;GET COUNT VALUE
69 006552 106622 MTPD (R2)+ ;MOVE TO USER'S AREA
70 006554 063704 000000G ADD PMNBPC,R4 ;ADVANCE ADDRESS COUNTER
71 006560 020437 000000G CMP R4,PMTOP ;RETURNED ALL VALUES?
72 006564 103003 BHIS 2$ ;BR IF YES
73 006566 077310 SOB R3,1$ ;CONTINUE MOVING
74 006570 005237 006614' INC PMOVFR ;SIGNAL OVERFLOW OF USER'S AREA
75 006574 000207 2$: RETURN ;EXIT SYSTEM STATE -- GO TO 3$
76 ;
77 ; Finished
78 ;
79 006576 013700 006614' 3$: MOV PMOVFR,R0 ;GET USER OVERFLOW ERROR FLAG
80 006602 001402 BEQ 4$ ;BRANCH IF NO OVERFLOW
81 006604 000137 000000G JMP SETERR ;ELSE REPORT ERROR TO USER
82 006610 000137 000000G 4$: JMP EMTXIT ;FINISHED
83 006614 000000 PMOVFR: .WORD 0 ;USER BUFFER OVERFLOW ERROR FLAG
84 ;
85 ;-----
86 ; Subroutine to check if the current user is doing a performance monitor.
87 ; If not an error return with code 0 is taken.
88 ;
89 006616 123737 000000G 000000G PMCHKU: CMPB CORUSR,PMUSER ;ARE WE PM USER?
90 006624 001410 BEQ 1$ ;BR IF YES
91 006626 005000 CLR R0 ;RETURN ERROR CODE 0
92 006630 113702 000000G MOVB PMUSER,R2 ;RETURN # OF USER WHO OWNS PM IN R0
93 006634 006202 ASR R2
94 006636 010237 000000G MOV R2,URO
95 006642 000137 000000G JMP SETERR
96 006646 000207 1$: RETURN
    
```

```

1          .SBTTL  CKACJB -- See if access to another job is allowed
2          ;-----
3          ; Determine if the current job is privileged to access another job.
4          ;
5          ; Inputs:
6          ;   R2 = Job index number of job we wish to access.
7          ;
8          ; Outputs:
9          ;   C-flag cleared ==> Access ok
10         ;   C-flag set ==> Access disallowed
11         ;
12 006650 010146  CKACJB: MOV      R1, -(SP)
13         ;
14         ; Always allow access to our own job
15         ;
16 006652 120237 000000G  CMPB    R2, CORUSR      ; Affecting our own job?
17 006656 001437      BEQ     7$              ; Br if yes
18         ;
19         ; If we have WORLD privilege we can access any job
20         ;
21 006660 032737 000000G 000000G  BIT    #P2$WRL, PRIVC2  ; Do we have WORLD privilege?
22 006666 001033      BNE    7$              ; Br if yes
23         ;
24         ; Always allow access to our virtual lines and children jobs.
25         ;
26 006670 113701 000000G      MOVB   CORUSR, R1      ; Get our job index number
27 006674 126162 000000G 000000G  CMPB   LNPRIM(R1), LNPRIM(R2) ; Do we have the same primary line?
28 006702 001425      BEQ     7$              ; Br if yes
29 006704 026201 000000G      CMP    LPARNT(R2), R1    ; Are we parent of this job?
30 006710 001422      BEQ     7$              ; Br if yes
31         ;
32         ; See if project numbers of jobs match
33         ;
34 006712 026162 000000G 000000G  CMP    LPROJ(R1), LPROJ(R2) ; Do project numbers match?
35 006720 001014      BNE    6$              ; Br if not -- We cannot change job
36 006722 032737 000000G 000000G  BIT    #P2$GRP, PRIVC2  ; Do we have GROUP privilege?
37 006730 001012      BNE    7$              ; Br if yes
38         ;
39         ; Project numbers match, check programmer numbers.
40         ;
41 006732 026162 000000G 000000G  CMP    LPROG(R1), LPROG(R2) ; Do programmer numbers match?
42 006740 001004      BNE    6$              ; Br if not
43 006742 032737 000000G 000000G  BIT    #P2$SAM, PRIVC2  ; Do we have SAME privilege?
44 006750 001002      BNE    7$              ; Br if yes
45         ;
46         ; We cannot access the job
47         ;
48 006752 000261 6$:      SEC                      ; Signal failure on return
49 006754 000401      BR     9$
50         ;
51         ; We can access the job
52         ;
53 006756 000241 7$:      CLC                      ; Signal success on return
54         ;
55         ; Finished
56         ;
57 006760 012601 9$:      MOV    (SP)+, R1

```

58 006762 000207
59 000001

RETURN
.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

\$BBIT	1-78	28-100						
\$AUTO	1-77	12-7	28-67					
\$CTRL0	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					
\$RNIOF	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						

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\$BBT	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							
DECHK	17-66#	17-97							
DETEM	1-43	17-10#							
DETERR	17-62#								
DETO	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			
DETEC	17-18	17-96#	17-99						
DFJMEM	1-83	3-39							

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#ITOP	1-62	3-153*																	
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							

... CM1	3-66												
... CM2	3-66	3-66	3-66	3-66		6-30							
... CM3	3-74	3-221											
... CM5	3-66	6-30											
... CM6	6-30												
... CM7	3-66												
.CLOSE	1-39#	3-74	3-221										
.CRRG	1-39#	6-30											
.READW	1-39#	3-66											
DIE	1-105#												
DISABL	1-90#	6-48											
ENABL	1-94#	6-53											
OCALL	1-116#	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								