

## Table of contents

2-	1	SETPRI -- Set priority or privilege for job
4-	1	KMCPCX -- Kmon EMT to copy context block data
5-	1	CPYEMT -- Copy file/privilege info from another job
6-	1	CPYCXD -- EMT to get information from another jobs context
7-	1	CPYFIL -- Copy file access info from another job
8-	1	CPYPRV -- Copy privilege flags from another job's context
9-	1	CPYCXT -- Copy info from another job's context block
10-	1	PRGLDC -- Purge channels opened to logical disks
11-	1	Job monitoring EMT's
18-	1	GETCXT -- Obtain exclusive access to context buffer
18-	39	FRECXT -- Free the context block buffer
19-	1	GETCHA -- Get address of user's channel block
20-	1	REDRMN -- Access simulated RMON for a job
21-	1	GTUKBL -- Access region used for user key definitions
22-	1	REDCXT -- Read a job context block into buffer
23-	1	CXBSWP -- Read job context data from swap file
24-	1	.CSISPC & .CSIGEN
29-	1	*** Subroutines ***
29-	2	CSIGET -- Get next character from CSI command line
30-	1	ACRFIL -- Accrue a file specification
31-	1	GTRD50 -- Accrue a RAD50 name
32-	1	R5OCHR -- Convert ascii character to Rad50 value
33-	1	SWVAL -- Accrue CSI switch value
34-	1	ACRDEC -- Accrue a decimal value
35-	1	ACROCT -- Accrue an octal value
36-	1	UPUSH -- Push value onto user's stack

1  
2  
3  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56  
57

```

        .TITLE  TSEM3  TSX-Plus EMT Overlay
        .ENABL  LC
        .ENABL  AMA
        .DSABL  GBL
;
; Copyright (C) 1976, 1977, 1978, 1979, 1980, 1981, 1982, 1983, 1984, 1985.
;
; S&H Computer Systems, Inc.
; Nashville, Tennessee
;
; This software is furnished under a license for use only
; on a single computer system and may be copied only with
; the inclusion of the above copyright notice.  This
; software, or any other copies thereof, may not be provided
; or otherwise made available to any other person except
; for use on such system and to one who agrees to these
; license terms.  Title to and ownership of the software
; shall at all times remain with S&H Computer Systems, Inc.
;
        .CSECT  TSEM3
TSEM2:  .RAD50  /EM3/           ;Overlay id
PS      =      177776         ;Processor Status Word
;
; Macro calls
;
        .MCALL  .CLOSE, .DSTATUS, .ENTER, .LOOKUP, .PURGE, .READW
;
; Global definitions
;
        .GLOBL  CSISPC, CSIGEN, MONEMT, MONABT, GETCXT, FRECX, GTUKBL
        .GLOBL  GETCHA, SETPRI, CPYEMT, KMPCPX, REDCXT
;
; Global references
;
        .GLOBL  PRIVFO, $VNOTT, SPPRED, SMONHD
        .GLOBL  NLCHN, C, CSW, CS$OPN, CS$NMX, LDDEVX, KEYRCB, RC$$SZ
        .GLOBL  RC$BLK, RC$BAS, LCXPAR, JPWDEV, JPWTYP, JPWFLG
        .GLOBL  NUCHN, STOP, ASNTBL, MAXASN, RESDEV, OKFILE, MAXACC
        .GLOBL  OF$$SZ, LDNAME, MAXLD, LDPDEV, LDSIZE, LDBASE, LDFLAG, CPYMNT
        .GLOBL  P2$VIR, PRIVC2, PO$SYS, $DILUP, LPARNT, P2$CXT, AT$$SZ
        .GLOBL  PRIVCO, PRIVSO, PVNPW, PO$SPV, PRIVAO, VPRIDF, MXJPRI, LBSPRI
        .GLOBL  SPCFLG, EMTBLK, URO, VALADW, CSIBUF, GETUCH, CFPNT, USRUCA
        .GLOBL  OVRHC, ASKLIN, VALADB, PUTUCH, EMTXIT, CSIEQL, SWTCNT, CSIDEV
        .GLOBL  CSIARE, ERRLOC, CSIMSG, SETERR, TAB, CSIFIL, CSIUSP, $NOVLN
        .GLOBL  LMONHD, MONFQH, JM$LNK, JM$JOB, JM$RTN, LSW9, LSW2, LSW2S
        .GLOBL  BADEMT, LSTSL, SETERR, UMODE, EMTPS, $INKMN, LSW4
        .GLOBL  GETQ, CQ$RO, CQ$R1, CQ$JOB, JM$RTN, CQ$RTN, CQ$CP, CP$STD
        .GLOBL  LPRI, CQ$PRI, S$IOFN, CQ$RNS, LITIME, S$HICP, QCOMPL
        .GLOBL  USWPCH, CXTBUF, CXBOWN, CXBJOB, CXBBAS, CXBSIZ
        .GLOBL  CORUSR, S$QCXB, QNSPNX, CHKABT, UREGD, $INCOR, LSW, CXBMOV
        .GLOBL  CXTBAS, LSWPBK, USREMT, PR7, INTPRI, CXTRMN
        .GLOBL  R$CHN, R$XCHN, CHNSIZ, MVSIZ
;-----
; Macros to enable and disable interrupts.
;
        .MACRO  DISABL           ;DISABLE INTERRUPTS

```

```

58          BIS      #340, @#PS
59          .ENDM    DISABL
60          ;
61          .MACRO   ENABL          ;ENABLE INTERRUPTS
62          BIC      INTPRI, @#PS
63          .ENDM    ENABL
64          ;
65          ; Macro to print an error message when a system crash occurs.
66          ;
67          ; Arguments:
68          ; MSG = Name of error message to print.
69          ; ARG = (Optional) argument value to display with error message.
70          ;
71          .GLOBL   DIEMSG, DIEARG, SYSHLT
72          .MACRO   DIE      MSG, ARG
73          MOV      MSG, @#DIEMSG
74          .IF      NB, ARG
75          MOV      ARG, @#DIEARG
76          .ENDC
77          CALL     @#SYSHLT
78          .ENDM    DIE
79          ;
80          ; Macro definition for calling global routines residing in mapped
81          ; system regions.
82          ;
83          .MACRO   OCALL     ENTADD
84          .IF      B, ENTADD
85          .ERROR   ;OCALL SPECIFIED WITH NO ENTRY ADDRESS
86          .MEXIT
87          .ENDC
88          CALL     OVRHC          ;CALL THE OVERLAY HANDLER
89          .WORD    ENTADD        ;SPECIFY THE ENTRY POINT
90          .ENDM
91          ;-----
92          ; Data areas
93          ;
94          000002 015270
          R50DK: .RAD50 /DK /

```

```

1          .SBTTL  SETPRI -- Set priority or privilege for job
2          ;-----
3          ; Set execution priority or privilege flags for job.
4          ;
5 000004 105737 000000G  SETPRI: TSTB  EMTBLK      ;Set priority or privilege?
6 000010 001027          BNE    SETPRV      ;Set privilege
7          ;
8          ; Set job execution priority
9          ;
10 000012 013702 000002G      MOV    EMTBLK+2,R2    ;Get specified execution priority
11 000016 003002          BGT    1$                ;Br if value specified
12 000020 113702 000000G      MOVB  VPRIDF,R2      ;If not specified, set default
13 000024 120237 000000G  1$:  CMPB  R2,MXJPRI    ;Compare with max authorized priority
14 000030 101402          BLOS  2$                ;Br if ok
15 000032 113702 000000G      MOVB  MXJPRI,R2      ;Get max authorized priority
16 000036 110261 000000G  2$:  MOVB  R2,LBSPRI(R1) ;Set base priority for job
17 000042 110261 000000G      MOVB  R2,LPRI(R1)   ;Set current priority for job
18 000046 032761 000000G 000000G  BIT    #$VNOTT,LSW(R1) ;Is this a subprocess not connected to term?
19 000054 001403          BEQ    9$                ;Br if not
20 000056          OCALL  SPPRED      ;Reduce priority of disconnected subprocesses
21 000064 000137 000000G  9$:  JMP    EMTXIT      ;Finished
  
```

SETPRI -- Set priority or privilege for job

```

1 ;-----
2 ; Set privilege flags for job.
3 ;
4 ; The form of the EMT argument block is as follows:
5 ;
6 ; .BYTE 1,150
7 ; .BYTE fun,perm ;Fun: 0=get,1=clear,2=set Perm: 0=temp,1=perm
8 ; .WORD buffer ;Buffer address
9 ; .WORD 0 ;(reserved)
10 ;
11 000070 SETPRV:
12 ;
13 ; Validate the user's buffer address
14 ;
15 000070 013702 0000040 MOV EMTBLK+4,R2 ;Get user's buffer address
16 000074 010200 MOV R2,R0
17 000076 004737 0000000 CALL VALADW ;Validate the address
18 ;
19 ; Only allow changes to authorization privilege flags if SETPRV is enabled.
20 ;
21 000102 123727 0000030 000002 CMPB EMTBLK+3,#2 ;Does he want to access auth privileges?
22 000110 001012 BNE 10$ ;Br if not
23 000112 032737 0000000 0000000 BIT #P0$SPV,PRIVCO ;Does he have SETPRV privilege?
24 000120 001006 BNE 10$ ;Br if yes
25 000122 123727 0000020 000002 CMPB EMTBLK+2,#2 ;Does he want to set bits?
26 000130 001002 BNE 10$ ;Br if not (allow access or clear-bits)
27 000132 105337 0000030 DECB EMTBLK+3 ;Can't change auth priv without SETPRV
28 ;
29 ; See what the subfunction is
30 ;
31 000136 113700 0000020 10$: MOVB EMTBLK+2,R0 ;Get subfunction code
32 000142 020027 000001 CMP R0,#1 ;Clear bits?
33 000146 001424 BEQ 2$ ;Br if yes
34 000150 020027 000002 CMP R0,#2 ;Set bits?
35 000154 001453 BEQ 5$ ;Br if yes
36 ;
37 ; Return current privilege flags to user
38 ;
39 000156 012703 0000000 MOV #PRIVCO,R3 ;Point to vector with current flags
40 000162 113700 0000030 MOVB EMTBLK+3,R0 ;Get type of priv flags wanted
41 000166 001406 BEQ 11$ ;Br if wants current priv flags
42 000170 012703 0000000 MOV #PRIVSO,R3 ;Point to table of perm flags
43 000174 005300 DEC R0 ;Want perm flags?
44 000176 001402 BEQ 11$ ;Br if yes
45 000200 012703 0000000 MOV #PRIVA0,R3 ;Must want authorized privileges
46 000204 012700 0000000 11$: MOV #PVNPW,R0 ;Get # words to move
47 000210 012346 1$: MOV (R3)+,-(SP) ;Get word with current flags
48 000212 106622 MTPD (R2)+ ;Move to user's buffer
49 000214 077003 SOB R0,1$ ;Move all words with privilege flags
50 000216 000504 BR 9$
51 ;
52 ; User wants to clear some privilege flags
53 ;
54 000220 005005 2$: CLR R5 ;Init index into privilege words
55 000222 113704 0000030 MOVB EMTBLK+3,R4 ;Get type of privilege change wanted
56 000226 106522 3$: MFPD (R2)+ ;Get flags from user
57 000230 012600 MOV (SP)+,R0 ;Pop flags

```

SETPRI -- Set priority or privilege for job

```

58 000232 040065 0000000 BIC R0,PRIVC0(R5) ;Clear current flags
59 000236 005704 TST R4 ;Change perm flags?
60 000240 001411 BEQ 4$ ;Br if not
61 000242 040065 0000000 BIC R0,PRIVS0(R5) ;Change permanent flags too
62 000246 040065 0000000 BIC R0,PRIVF0(R5) ;Change command file privileges
63 000252 020427 0000002 CMP R4,#2 ;Change auth flags?
64 000256 001002 BNE 4$ ;Br if not
65 000260 040065 0000000 BIC R0,PRIVA0(R5) ;Change authorization flags too
66 000264 062705 0000002 4$: ADD #2,R5 ;Get index to next priv word
67 000270 020527 0000000 CMP R5,#PVNPW*2 ;Done all words?
68 000274 103754 BLO 3$ ;Loop if not
69 000276 004737 000434' CALL FIXPRV ;Update LSW tables
70 000302 000452 BR 9$
71 ;
72 ; User wants to set some privilege flags
73 ;
74 000304 005001 5$: CLR R1 ;Say no error detected yet
75 000306 005005 CLR R5 ;Init index into priv words
76 000310 106522 8$: MFPD (R2)+ ;Get flags to be set
77 000312 012604 MOV (SP)+,R4 ;Get flags
78 000314 032737 0000000 0000000 BIT #P0$SPV,PRIVC0 ;Is user allowed to set any flag?
79 000322 001006 BNE 6$ ;Br if yes
80 000324 010400 MOV R4,R0 ;Get flags to set
81 000326 046500 0000000 BIC PRIVA0(R5),R0 ;Is user allowed to set all of these flags?
82 000332 001402 BEQ 6$ ;Br if yes
83 000334 040004 BIC R0,R4 ;Clear disallowed flags
84 000336 005201 INC R1 ;Remember to report error
85 000340 050465 0000000 6$: BIS R4,PRIVC0(R5) ;Set current flags
86 000344 105737 0000003G TSTB EMTBLK+3 ;Is change to be permanent?
87 000350 001412 BEQ 7$ ;Br if not
88 000352 050465 0000000 BIS R4,PRIVS0(R5) ;Make permanent change
89 000356 050465 0000000 BIS R4,PRIVF0(R5) ;Change command file privileges
90 000362 123727 0000003G 0000002 CMPB EMTBLK+3,#2 ;Change auth flags too?
91 000370 001002 BNE 7$ ;Br if not
92 000372 050465 0000000 BIS R4,PRIVA0(R5) ;Change authorization privileges
93 000376 062705 0000002 7$: ADD #2,R5 ;Get index to next priv word
94 000402 020527 0000000 CMP R5,#PVNPW*2 ;More priv words to change?
95 000406 103740 BLO 8$ ;Br if yes
96 000410 004737 000434' CALL FIXPRV ;Update LSW tables
97 000414 005701 TST R1 ;Do we need to report an error?
98 000416 001404 BEQ 9$ ;Br if not
99 000420 012700 0000001 MOV #1,R0 ;Return error code 1
100 000424 000137 0000000 JMP SETERR
101 ;
102 ; Finished
103 ;
104 000430 000137 0000000 9$: JMP EMTXIT ;Finished
105 ;
106 ;-----
107 ; FIXPRV is called after changing any privilege flags to set any
108 ; privilege flags in LSW tables.
109 ;
110 000434 010146 FIXPRV: MOV R1,-(SP)
111 000436 113701 0000000 MOVB CORUSR,R1 ;Get current job index number
112 ;
113 ; See if virtual line access is allowed?
114 ;

```

```
115 000442 042761 0000000 0000000      BIC    ##NOVLN,LSW2(R1);Assume virtual line access is allowed
116 000450 042761 0000000 0000000      BIC    ##NOVLN,LSW2S(R1)
117 000456 032737 0000000 0000000      BIT    #P2$VIR,PRIVC2 ;Is virtual line access allowed?
118 000464 001006      BNE    1$ ;Br if yes
119 000466 052761 0000000 0000000      BIS    ##NOVLN,LSW2(R1);Disable virtual line access
120 000474 052761 0000000 0000000      BIS    ##NOVLN,LSW2S(R1)
121                                     ;
122                                     ; Finished
123                                     ;
124 000502 012601      1$:    MOV    (SP)+,R1
125 000504 000207      RETURN
```

```
1 .SBTTL KMPCPX -- Kmon EMT to copy context block data
2 ;-----
3 ; Tskmon EMT to copy data from the context block of another job into the
4 ; context block of our job.
5 ;
6 ; Form of EMT argument block:
7 ;
8 ; .BYTE 0,126
9 ; .WORD 14
10 ; .WORD job_number
11 ; .WORD address_of_item
12 ; .WORD num_bytes_to_copy
13 ;
14 000506 KMPCPX:
15 ;
16 ; Gain exclusive access to context block buffer
17 ;
18 000506 004737 002324' CALL GETCXT ;Gain access to context buffer
19 000512 103002 BCC 1$ ;Br if got it
20 000514 004737 000000G CALL STOP ;Job was aborted
21 ;
22 ; Copy the requested data
23 ;
24 000520 013702 000004G 1$: MOV EMTBLK+4,R2 ;Get # of job we are copying from
25 000524 013703 000006G MOV EMTBLK+6,R3 ;Get address of item we are copying
26 000530 013705 000010G MOV EMTBLK+8,R5 ;Get # bytes to copy
27 000534 004737 001342' CALL CPYCXT ;Copy the data
28 ;
29 ; Finished
30 ;
31 000540 004737 002410' CALL FREEXT ;Free the context buffer
32 000544 000137 000000G JMP EMTXIT
```

```

1          .SBTTL  CPYEMT -- Copy file/privilege info from another job
2          ;-----
3          ; This EMT is used to copy the file and privilege information from another
4          ; job.  The form of the EMT argument block is:
5          ;
6          ;       .BYTE   subfun,160
7          ;       .WORD   jobnum
8          ;       .WORD   0
9          ;
10         ; Where subfun has the following meanings:
11         ; 0 ==> Copy file assess information only.
12         ; 1 ==> Copy file info and privilege info.
13         ; 3 ==> Copy info from context block into program buffer.
14         ; Jobnum is the number of the job that we are copying from.
15         ;
16 000550  CPYEMT:
17         ;
18         ; See if the specified job is logged on
19         ;
20 000550  013702  0000020      MOV     EMTBLK+2,R2      ;Get # of job we are copying from
21 000554  006302              ASL     R2                ;Convert to job index number
22 000556  001407              BEQ     10$                ;Br if job number 0
23 000560  020227  0000000      CMP     R2,#LSTSL        ;Valid job number?
24 000564  101004              BHI     10$                ;Br if not
25 000566  032762  0000000  0000000  BIT     ##DILUP,LSW(R2) ;Is job logged on?
26 000574  001004              BNE     1$                ;Br if yes
27 000576  012700  0000001      10$:  MOV     #1,R0          ;Return error code 1
28 000602  000137  0000000      JMP     SETERR
29         ;
30         ; See if job is privileged to do this
31         ;
32 000606  026102  0000000      1$:  CMP     LPARNT(R1),R2 ;Are we copying info from our parent job?
33 000612  001407              BEQ     4$                ;Br if yes -- That is always legal
34 000614  032737  0000000  0000000  BIT     #P2$CXT,PRIVC2 ;Are we authorized to copy context info?
35 000622  001003              BNE     4$                ;Br if yes
36 000624  005000              3$:  CLR     R0                ;Return error code 0
37 000626  000137  0000000      JMP     SETERR
38         ;
39         ; See if we are copying into our context block or into program buffer
40         ;
41 000632  123727  0000000  0000003  4$:  CMPB    EMTBLK,#3        ;Copy data into program buffer?
42 000640  001002              BNE     2$                ;Br if not
43 000642  000137  000676'      JMP     CPYCXD            ;Copy context data into program buffer
44         ;
45         ; Copy the file access info
46         ;
47 000646  004737  001022'      2$:  CALL    CPYFIL            ;Copy file access info for job
48         ;
49         ; Copy the privilege info
50         ;
51 000652  105737  0000000      TSTB   EMTBLK            ;Does he want to copy privileges?
52 000656  001405              BEQ     9$                ;Br if not
53 000660  026102  0000000      CMP     LPARNT(R1),R2    ;Copying privileges from parent?
54 000664  001357              BNE     3$                ;Br if not
55 000666  004737  001222'      CALL    CPYPRV            ;Copy privilege info
56         ;
57         ; Finished

```

58 ;  
59 000672 000137 000000G 9#: JMP EMTXIT ;Finished with EMT

```

1          .SBTTL  CPYCXD --- EMT to get information from another jobs context
2          ;-----
3          ; Copy data from the context block of a job into a program buffer area.
4          ;
5          ; Inputs:
6          ; R1 = Index number of our job
7          ; R2 = Index number of job we are copying from
8          ;
9          ; EMT argument block:
10         ;
11         ; .BYTE 3,160
12         ; .WORD job_number
13         ; .WORD item_index
14         ; .WORD buffer_address
15         ;
16 000676 CPYCXD:
17         ;
18         ; Gain exclusive access to context buffer
19         ;
20 000676 004737 002324'      CALL  GETCXT      ;Gain access to context buffer
21 000702 103002             BCC   1$             ;Br if got it
22 000704 004737 000000G     CALL  STOP        ;Aborted while waiting
23         ;
24         ; Get info about data to be copied
25         ;
26 000710 013704 000004G     1$:  MOV   EMTBLK+4,R4      ;Get index # of item of interest
27 000714 006304             ASL   R4                ;Convert to word table index
28 000716 020427 000004     CMP   R4,#MAXCXX      ;Is it too big?
29 000722 101404             BLOS  2$             ;Br if ok
30 000724 012700 000002     MOV   #2,R0         ;Return error 2 if too big
31 000730 000137 000000G     JMP   SETERR
32 000734 016403 001006'     2$:  MOV   CXXADR(R4),R3    ;Get address of start of data
33 000740 016404 001014'     MOV   CXXSIZ(R4),R4    ;Get # bytes wanted
34 000744 010400             MOV   R4,R0
35 000746 004737 002726'     CALL  REDCXT        ;Read data into context buffer
36         ;
37         ; Move data to user's buffer
38         ;
39 000752 013700 000006G     MOV   EMTBLK+6,R0     ;Get address of user's buffer
40 000756 004737 000000G     CALL  VALADW        ;Make sure buffer is ok
41 000762 013705 000000G     MOV   CXTBUF,R5      ;Get address of buffer
42 000766 006204             ASR   R4                ;Get # words to move
43 000770 012546             3$:  MOV   (R5)+,-(SP)      ;Get next word
44 000772 106620             MTPD  (R0)+          ;Move to user's buffer
45 000774 077403             SOB   R4,3$          ;Loop if more to move
46         ;
47         ; Finished
48         ;
49 000776 004737 002410'     CALL  FRECXT        ;Free context buffer
50 001002 000137 000000G     JMP   EMTXIT
51         ;
52         ; Table of addresses of items in context block
53         ;
54 001006 000000G     CXXADR: .WORD  JPWDEV      ; 0 - Name of print-window device
55 001010 000000G     .WORD  JPWTYP        ; 1 - Print-window device type
56 001012 000000G     .WORD  JPWFLG        ; 2 - Print-window flag word
57 000004             MAXCXX = .-CXXADR-2

```

```
58 ;  
59 ; Number of bytes to move for each item  
60 ;  
61 001014 000002 CXXSIZ: .WORD 2 ; 0 - Name of print-window device  
62 001016 000002 .WORD 2 ; 1 - Print-window device type  
63 001020 000002 .WORD 2 ; 2 - Print-window flag word  
64 ;
```

CPYFIL -- Copy file access info from another job

```

1          .SBTTL  CPYFIL -- Copy file access info from another job
2          ;-----
3          ; CPYFIL is called to copy all of the file access information from
4          ; another job to our job.  The information copied is as follows:
5          ; 1. ASSIGN commands.
6          ; 2. Logical disk information.
7          ; 3. Mounted devices information.
8          ; 4. ACCESS control information.
9          ;
10         ; Inputs:
11         ; R2 = Job index number of job whose file info is to be copied.
12         ;
13 001022 010346 CPYFIL: MOV     R3,-(SP)
14 001024 010546      MOV     R5,-(SP)
15         ;
16         ; Purge all channels that are open to logical disks
17         ;
18 001026 004737 001372'      CALL    PRGLDC      ;Purge all LD channels
19         ;
20         ; Dismount all mounted devices for this job
21         ;
22 001032 012700 000000G      MOV     #CSIARE,R0      ;Point to internal EMT arg block area
23 001036 012710 056402      MOV     #2+<135*400>,(R0);Set EMT function code
24 001042 104375      EMT     375          ;Dismount all devices for job
25         ;
26         ; Gain exclusive access to job context buffer
27         ;
28 001044 004737 002324'      CALL    GETCXT      ;Get access to context buffer
29 001050 103002      BCC    3$          ;Br if got it
30 001052 004737 000000G      CALL    STOP        ;Job was aborted while waiting
31         ;
32         ; Set up ASSIGN information
33         ;
34 001056 012703 000000G 3$:  MOV     #ASNTBL,R3      ;Point to assign table
35 001062 012705 000000C      MOV     #MAXASN*AT*$SZ,R5;# bytes to get
36 001066 004737 001342'      CALL    CPYCXT      ;Copy data into our context block
37         ;
38         ; Set up ACCESS information
39         ;
40 001072 012703 000000G      MOV     #RESDEV,R3      ;Get this control word
41 001076 012705 000002      MOV     #2,R5          ;2 bytes for this
42 001102 004737 001342'      CALL    CPYCXT      ;Copy data into our context block
43 001106 012703 000000G      MOV     #DKFILE,R3     ;Now get actual access table
44 001112 012705 000000C      MOV     #MAXACC*OF*$SZ,R5 ;# bytes in table
45 001116 004737 001342'      CALL    CPYCXT      ;Copy data into our context block
46         ;
47         ; Set up logical disk information
48         ;
49 001122 012703 000000G      MOV     #LDNAME,R3     ;Copy each LD control table
50 001126 012705 000000C      MOV     #8.*MAXLD,R5
51 001132 004737 001342'      CALL    CPYCXT
52 001136 012703 000000G      MOV     #LDPDEV,R3
53 001142 012705 000000C      MOV     #2*MAXLD,R5
54 001146 004737 001342'      CALL    CPYCXT
55 001152 012703 000000G      MOV     #LDSIZE,R3
56 001156 004737 001342'      CALL    CPYCXT
57 001162 012703 000000G      MOV     #LDBASE,R3

```

```
58 001166 004737 001342'          CALL  CPYCXT
59 001172 012703 000000G          MOV   #LDFLAG,R3
60 001176 004737 001342'          CALL  CPYCXT
61                                     ;
62                                     ; Release the context block buffer
63                                     ;
64 001202 004737 002410'          CALL  FRECXT          ;Free job context buffer
65                                     ;
66                                     ; Copy mounts
67                                     ;
68 001206                                     OCALL CPYMNT          ;Copy mounts
69                                     ;
70                                     ; Finished
71                                     ;
72 001214 012605          MOV   (SP)+,R5
73 001216 012603          MOV   (SP)+,R3
74 001220 000207          RETURN
```

```

1          .SBTTL  CPYPRV -- Copy privilege flags from another job's context
2          ;-----
3          ; Copy the authorized and set privilege flags from another job's context
4          ; block to our context block.
5          ;
6          ; Inputs:
7          ; R2 = Index number of job we are copying from.
8          ;
9 001222 010346 CPYPRV: MOV     R3,-(SP)
10 001224 010546      MOV     R5,-(SP)
11          ;
12          ; Gain exclusive access to context buffer
13          ;
14 001226 004737 002324'      CALL    GETCXT      ;Gain access to context buffer
15 001232 103002      BCC     3$          ;Br if got it
16 001234 004737 0000000     CALL    STOP        ;Job was aborted
17          ;
18          ; Copy authorized privileges
19          ;
20 001240 012703 0000000     3$:   MOV     #PRIVA0,R3      ;Point to authorized privilege vector
21 001244 012705 0000000     MOV     #PVNPW*2,R5     ;Get # bytes to copy
22 001250 004737 001342'     CALL    CPYCXT        ;Copy context info
23          ;
24          ; Copy set privileges
25          ;
26 001254 012703 0000000     MOV     #PRIVS0,R3     ;Point to set privilege vector
27 001260 004737 001342'     CALL    CPYCXT        ;Copy context into
28          ;
29          ; Set current privileges to set privileges
30          ;
31 001264 012700 0000000     MOV     #PVNPW,R0     ;Get # privilege words
32 001270 005005      CLR     R5            ;Init index
33 001272 016565 0000000 0000000 1$:  MOV     PRIVS0(R5),PRIVC0(R5);Current privileges
34 001300 016565 0000000 0000000     MOV     PRIVS0(R5),PRIVF0(R5);Command file privileges
35 001306 062705 0000002     ADD     #2,R5        ;Increment word index
36 001312 077011      SOB     R0,1$        ;Loop if more to copy
37          ;
38          ; Copy maximum authorized priority
39          ;
40 001314 012703 0000000     MOV     #MXJPRI,R3   ;Get maximum authorized priority
41 001320 012705 0000001     MOV     #1,R5        ;1 byte
42 001324 004737 001342'     CALL    CPYCXT
43          ;
44          ; Finished
45          ;
46 001330 004737 002410'     CALL    FRECXT        ;Free context buffer
47 001334 012605      MOV     (SP)+,R5
48 001336 012603      MOV     (SP)+,R3
49 001340 000207      RETURN

```

```
1 .SBTTL CPYCXT -- Copy info from another job's context block
2 ;-----
3 ; CPYCXT is called to copy information from another job's context block
4 ; into our context block.
5 ;
6 ; Inputs:
7 ; R2 = Index number of job that we are copying from.
8 ; R3 = Address within context block of info to be copied.
9 ; R5 = Number of bytes to copy.
10 ;
11 001342 010346 CPYCXT: MOV R3,-(SP)
12 001344 010546 MOV R5,-(SP)
13 ;
14 ; Read the info into the context block buffer
15 ;
16 001346 010500 MOV R5,R0 ;Get # bytes to be read
17 001350 004737 002726' CALL REDCXT ;Move info into context buffer
18 ;
19 ; Now move info from buffer to our context block
20 ;
21 001354 013700 000000G MOV CXTBUF,R0 ;Point to buffer
22 001360 112023 1#: MOV (R0)+,(R3)+ ;Copy data into our context area
23 001362 077502 SOB R5,1#
24 ;
25 ; Finished
26 ;
27 001364 012605 MOV (SP)+,R5
28 001366 012603 MOV (SP)+,R3
29 001370 000207 RETURN
```

```

1          .SBTTL  PRGLDC -- Purge channels opened to logical disks
2          ;-----
3          ; PRGLDC is called to purge all channels that are opened to logical disks.
4          ;
5 001372 010346 PRGLDC: MOV      R3, -(SP)
6 001374 010546          MOV      R5, -(SP)
7          ;
8          ; Begin loop through all channels
9          ;
10 001376 012703 1777770          MOV      #NLCHN-1, R3          ;Get # of last channel
11          ;
12          ; Compute address of channel block for this channel
13          ;
14 001402 010305 1#:          MOV      R3, R5          ;Get channel number
15 001404 013700 0000000          MOV      CXTRMN, R0          ;Get address of base of simulated RMON
16 001410 062700 0000000          ADD      #R$CHN, R0          ;Point to block for channel 0
17 001414 020527 000021          CMP      R5, #21          ;Is this channel in extended channel area?
18 001420 103404          BLO      2$          ;Br if not
19 001422 162705 000021          SUB      #21, R5          ;Get channel # relative to extended channels
20 001426 062700 0000000          ADD      #R$XCHN-R$CHN, R0; Point to extended channel area
21 001432 070527 0000000 2#:          MUL      #CHNSIZ, R5          ;Multiply by # bytes per channel
22 001436 060005          ADD      R0, R5          ;Get address of channel block
23          ;
24          ; See if this channel is opened to a logical disk
25          ;
26 001440 016500 0000000          MOV      C.CSW(R5), R0          ;Get status word for channel
27 001444 032700 0000000          BIT      #CS$OPN, R0          ;Is this channel open?
28 001450 001411          BEQ      3$          ;Br if not
29 001452 042700 0000000          BIC      #^C<CS$NMX>, R0          ;Extract device index number
30 001456 120037 0000000          CMPB    R0, LDDEVX          ;Is chan opened to a logical disk?
31 001462 001004          BNE      3$          ;Br if not
32          ;
33          ; Purge this channel
34          ;
35 001464          .PURGE  R3          ;Purge the channel
36          ;
37          ; Check the next channel
38          ;
39 001474 005303 3#:          DEC      R3          ;Get next channel number
40 001476 002341          BGE      1$          ;Loop if more to check
41          ;
42          ; Finished
43          ;
44 001500 012605          MOV      (SP)+, R5
45 001502 012603          MOV      (SP)+, R3
46 001504 000207          RETURN

```

```

1          .SBTTL Job monitoring EMT's
2          ;-----
3          ; The following EMT's are part of the TSX-Plus facility that allows one
4          ; job to monitor the state of another job.
5          ;
6          ; The form of the EMT is:
7          ;
8          ;     .BYTE   code,157
9          ;     .WORD   arg1
10         ;     .WORD   arg2
11         ;
12         ; Where "code" is the sub-function code indicating which job-monitoring
13         ; function is wanted.
14         ;
15 001506  MONEMT:
16         ;
17         ; Get sub-function code and see if it is legal.
18         ;
19 001506  113702  0000009      MOVB   EMTBLK,R2      ;Get sub-function code
20 001512  006302              ASL    R2              ;Convert to word table index
21 001514  020227  000010      CMP    R2,#MXMONF    ;Is it legal?
22 001520  101402              BLOS  1$              ;Br if ok
23 001522  000137  0000009      JMP    BADEMT        ;Invalid sub-function code
24 001526  000172  001532'    1$:   JMP    @MONFUN(R2) ;Enter processing routine based on sub-functn
25         ;
26         ; Jump vector for sub-function codes
27         ;
28 001532  001542'    MONFUN: .WORD  MON0      ; 0 -- Begin monitoring a job
29 001534  001666'      .WORD  MON1      ; 1 -- Stop monitoring a job
30 001536  001750'      .WORD  MON2      ; 2 -- Send status to monitoring jobs
31 001540  001732'      .WORD  MON3      ; 3 -- Send status from system program
32         000010      MXMONF = .-MONFUN ;Max sub-function index
  
```

```

1 ;-----
2 ; Sub-function # 0
3 ; Begin monitoring a specified job.
4 ;
5 001542 MONO:
6 ;
7 ; Get the number of the job to monitor and see if it is valid
8 ;
9 001542 013702 000002G      MOV      EMTBLK+2,R2      ;Get # of job to be monitored
10 001546 006302           ASL      R2              ;Convert to job index number
11 001550 020227 000000G    CMP      R2,#LSTSL      ;Is the job number too large?
12 001554 101040           BHI      B$              ;Br if yes
13 ;
14 ; The job number is valid.
15 ; If our job has previously set up monitoring on the same job we are
16 ; about to monitor, release the previous monitoring.
17 ;
18 001556 004737 002234'    CALL     MONREL          ;Release any previous monitoring
19 ;
20 ; Get a free monitoring control block
21 ;
22 001562 013703 000000G    MOV      MONFGH,R3      ;Get a free monitor control block
23 001566 001004           BNE      2$              ;Br if got one
24 001570 012700 000002    MOV      #2,R0          ;Return error 2 if no free control blocks
25 001574 000137 000000G    JMP      SETERR
26 001600 016337 000000G 000000G 2$:  MOV      JM$LNK(R3),MONFGH ;Remove block from free list
27 ;
28 ; Initialize the monitoring control block
29 ;
30 001606 110163 000000G    MOVB     R1,JM$JOB(R3)   ;Set # of job that is monitoring
31 001612 013763 000004G 000000G  MOV      EMTBLK+4,JM$RTN(R3) ;Set address of completion routine
32 ;
33 ; Add monitoring control block to list for job being monitored
34 ;
35 001620 005702           TST      R2              ;Request to monitor all jobs?
36 001622 001004           BNE      3$              ;Br if not
37 001624 013763 000000G 000000G  MOV      SMONHD,JM$LNK(R3);Add to list of requests for all jobs
38 001632 010337 000000G    MOV      R3,SMONHD
39 001636 000405           BR       9$
40 001640 016263 000000G 000000G 3$:  MOV      LMONHD(R2),JM$LNK(R3) ;Add to list for job being monitored
41 001646 010362 000000G    MOV      R3,LMONHD(R2)
42 ;
43 ; Finished
44 ;
45 001652 000137 000000G 9$:  JMP      EMTXIT
46 ;
47 ; Error, invalid job number
48 ;
49 001656 012700 000001    8$:  MOV      #1,R0          ;Return error code 1
50 001662 000137 000000G    JMP      SETERR

```

```

1 ;-----
2 ; Subfunction 1.
3 ; Stop monitoring a job.
4 ;
5 001666 MONI:
6 ;
7 ; Get the number of the job to monitor and see if it is valid
8 ;
9 001666 013702 0000020      MOV     EMTBLK+2,R2      ;Get # of job to be monitored
10 001672 001407            BEQ     1$              ;Zero ==> Release all monitors
11 001674 006302            ASL     R2                ;Convert to job index number
12 001676 020227 0000000    CMP     R2,#LSTSL       ;Is the job number too large?
13 001702 101007            BHI     8$              ;Br if yes
14 ;
15 ; Release monitoring a specific job
16 ;
17 001704 004737 002234'    CALL   MONREL          ;Release monitoring the specified job
18 001710 000402            BR     9$
19 ;
20 ; Release monitoring all jobs
21 ;
22 001712 004737 002210'    1$:   CALL   MONABT          ;Release all job monitoring
23 ;
24 ; Finished
25 ;
26 001716 000137 0000000    9$:   JMP     EMTXIT
27 ;
28 ; Error -- Invalid job number specified
29 ;
30 001722 012700 000001    8$:   MOV     #1,R0          ;Return error code 1
31 001726 000137 0000000    JMP     SETERR

```

Job monitoring EMT's

```

1 ;-----
2 ; Subfunctions 2 and 3.
3 ; Send a signal to monitoring jobs.
4 ;
5 001732 032737 000000G 000000G MON3: BIT #PO$SYS,PRIVCO ;Do we have SYSPRV privilege?
6 001740 001003 BNE MON2 ;Br if yes
7 001742 112737 000002 000000G MOVB #2,EMTBLK ;Change to user send if not
8 001750 MON2:
9 ;
10 ; If there are no monitoring jobs, return error code 0.
11 ;
12 001750 016103 000000G MOV LMONHD(R1),R3 ;Are there any monitoring jobs?
13 001754 001006 BNE 1$ ;Br if yes
14 001756 005737 000000G TST SMONHD ;Any monitoring of all jobs
15 001762 001003 BNE 1$ ;Br if yes
16 001764 005000 CLR R0 ;Return error 0 -- Noone monitoring us
17 001766 000137 000000G JMP SETERR
18 ;
19 ; There are some monitoring jobs.
20 ; Determine if this status is being generated by the system or by
21 ; a program being run from the monitored line.
22 ;
23 001772 010105 1$: MOV R1,R5 ;Get our job index number
24 001774 072527 177777 ASH #-1,R5 ;Convert to job number
25 002000 032737 000000G 000000G BIT #UMODE,EMTPS ;Was EMT done in user or kernel mode?
26 002006 001412 BEQ 3$ ;Br if in kernel mode
27 002010 032761 000000G 000000G BIT ##INKMN,LSW4(R1); Is TSKMON sending status?
28 002016 001006 BNE 3$ ;Br if yes
29 002020 123727 000000G 000003 CMPB EMTBLK,#3 ;Send status from system program (like LOGON)
30 002026 001402 BEQ 3$ ;Br if yes
31 002030 052705 100000 BIS #100000,R5 ;Set user-generated-status flag
32 ;
33 ; Begin loop to generate a completion routine for each job that is
34 ; monitoring us.
35 ;
36 002034 005703 3$: TST R3 ;Any requests to monitor this specific job?
37 002036 001405 BEQ 5$ ;Br if not
38 002040 004737 002076' 6$: CALL MONQUE ;Queue a completion routine for monitoring job
39 002044 016303 000000G MOV JM$LNK(R3),R3 ;Are there more monitoring jobs?
40 002050 001373 BNE 6$ ;Loop if yes
41 ;
42 ; See if there are some pending monitoring requests for all jobs
43 ;
44 002052 013703 000000G 5$: MOV SMONHD,R3 ;Any requests to monitor all jobs?
45 002056 001405 BEQ 9$ ;Br if not
46 002060 004737 002076' 4$: CALL MONQUE ;Queue the completion routine
47 002064 016303 000000G MOV JM$LNK(R3),R3 ;More requests?
48 002070 001373 BNE 4$ ;Loop if yes
49 ;
50 ; Finished
51 ;
52 002072 000137 000000G 9$: JMP EMTXIT

```

```

1 ;-----
2 ; MONQUE is called to queue a completion routine to pass a monitoring
3 ; status message to a monitoring job.
4 ;
5 ; Inputs:
6 ; R3 = Pointer to monitor control block (JM$xxx)
7 ; R5 = Job number and system status flag to be passed in R0
8 ; EMTBLK+2 = Status code to be passed to completion routine in R1
9 ;
10 002076 010146 MONQUE: MOV R1,-(SP)
11 002100 010446 MOV R4,-(SP)
12 ;
13 ; Get a free completion queue element.
14 ;
15 002102 004737 000000G CALL GETQ ;Get free queue element (addr in R1)
16 ;
17 ; Set up information in the completion queue element
18 ;
19 002106 010561 000000G MOV R5,CQ$R0(R1) ;Pass job number in R0
20 002112 013761 000002G 000000G MOV EMTBLK+2,CQ$R1(R1) ;Pass status code in R1
21 002120 116300 000000G MOVB JM$JOB(R3),R0 ;Get # of job who is monitoring us
22 002124 110061 000000G MOVB R0,CQ$JOB(R1) ;Completion routine is for that job
23 002130 016361 000000G 000000G MOV JM$RTN(R3),CQ$RTN(R1);Set address of completion routine
24 002136 116061 000000G 000000G MOVB LPRI(R0),CQ$PRI(R1);Set execution priority
25 002144 112761 000000G 000000G MOVB #S$IOFN,CQ$RNS(R1);Set job execution state
26 002152 112761 000000G 000000G MOVB #CP$STD,CQ$CP(R1);Set compl routine class priority
27 002160 005760 000000G TST LITIME(R0) ;Is this an interactive job?
28 002164 001403 BEQ Z$ ;Br if not
29 002166 112761 000000G 000000G MOVB #S$HICP,CQ$RNS(R1);Set interactive state
30 002174 010104 Z$: MOV R1,R4 ;Get address of Q element to R4 for QCOMPL
31 ;
32 ; Queue a completion routine for the monitoring job
33 ;
34 002176 004737 000000G CALL QCOMPL ;Queue a completion routine
35 ;
36 ; Finished
37 ;
38 002202 012604 MOV (SP)+,R4
39 002204 012601 MOV (SP)+,R1
40 002206 000207 RETURN
  
```

```
1 ;-----  
2 ; MONABT is called to release all job monitoring requests issued by a  
3 ; job.  
4 ;  
5 ; Inputs:  
6 ; R1 = Job index number of job whose monitoring requests are to be released.  
7 ;  
8 002210 010246 MONABT: MOV R2, -(SP)  
9 ;  
10 ; Begin loop to release requests for all jobs on the system.  
11 ;  
12 002212 012702 0000000 MOV #LSTSL, R2 ;Get # of last line  
13 ;  
14 ; Release all monitoring being done on that job by our job  
15 ;  
16 002216 004737 002234' 1$: CALL MONREL ;Release monitoring  
17 ;  
18 ; Loop if more jobs to check  
19 ;  
20 002222 162702 000002 SUB #2, R2 ;Are there more jobs to check?  
21 002226 002373 BGE 1$ ;Loop if yes (0=monitor all jobs)  
22 ;  
23 ; Finished  
24 ;  
25 002230 012602 MOV (SP)+, R2  
26 002232 000207 RETURN
```

```

1 ;-----
2 ; MONREL is called to release a monitoring request that has been
3 ; made by one job on another job.
4 ;
5 ; Inputs:
6 ; R1 = Job index of job that issued the monitor request.
7 ; R2 = Job index of the job that is being monitored (0=all jobs).
8 ;
9 002234 010346 MONREL: MOV R3, -(SP)
10 ;
11 ; If monitored job # = 0 then check system monitor chain,
12 ; otherwise check chain for monitored job.
13 ;
14 002236 005702 TST R2 ;Monitoring 1 job or all jobs?
15 002240 001003 BNE 3$ ;Br if monitoring 1 job
16 002242 012703 0000000 MOV #SMONHD, R3 ;Point to list head for sys monitor list
17 002246 000403 BR 1$
18 ;
19 ; Begin to follow chain of monitoring blocks for job being monitored
20 ;
21 002250 012703 0000000 3$: MOV #LMONHD, R3 ;Point to list head for monitor blocks
22 002254 060203 ADD R2, R3 ;Point to list head for job of interest
23 002256 011300 1$: MOV (R3), R0 ;Get address of next monitor control block
24 002260 001417 BEQ 9$ ;Br if no more
25 002262 120160 0000000 CMPB R1, JM$JOB(R0) ;Was this block issued by job of interest?
26 002266 001010 BNE 2$ ;Br if not
27 002270 016013 0000000 MOV JM$LNK(R0), (R3) ;Remove block from list
28 002274 013760 0000000 0000000 MOV MONFQH, JM$LNK(R0);Return block to free list
29 002302 010037 0000000 MOV R0, MONFQH
30 002306 000763 BR 1$
31 002310 010003 2$: MOV R0, R3 ;Get address of next block
32 002312 062703 0000000 ADD #JM$LNK, R3 ;Point to cell with forward link
33 002316 000757 BR 1$ ;Go continue following list
34 ;
35 ; Finished
36 ;
37 002320 012603 9$: MOV (SP)+, R3
38 002322 000207 RETURN
  
```

GETCXT -- Obtain exclusive access to context buffer

```

1          .SBTTL  GETCXT -- Obtain exclusive access to context buffer
2          ;-----
3          ; This routine is called to obtain exclusive access to the buffer
4          ; used to access the context block for another job (CXTBUF).
5          ; On return, exclusive access has been granted to the context
6          ; block buffer unless the job was aborted while waiting for access.
7          ;
8          ; Outputs:
9          ;   C-flag cleared ==> Access granted.
10         ;   C-flag set ==> Job aborted while waiting for buffer.
11         ;
12 002324  GETCXT:
13         ;
14         ; See if the context buffer is currently free
15         ;
16 002324  1$:   DISABL          ;;;Disable interrupts
17 002332  113700 0000000  MOVB    CXBOWN,RO      ;;;Is buffer currently free?
18 002336  001415          BEQ     3$          ;;;Br if yes
19 002340  120037 0000000  CMPB   RO,CORUSR      ;;;Do we already own buffer?
20 002344  001415          BEQ     2$          ;;;Br if yes
21         ;
22         ; Someone else owns the buffer now.
23         ; Suspend our job until we can get it.
24         ;
25 002346          ENABL          ;Enable interrupts
26 002354  012700 0000000  MOV    #S$QCXB,RO    ;Get waiting-for-buffer wait state
27 002360  004737 0000000  CALL   QNSPNX        ;Suspend job until buffer is available
28 002364  004737 0000000  CALL   CHKABT        ;Were we aborted while asleep?
29 002370  000755          BR     1$          ;Go try to get buffer
30         ;
31         ; We can get buffer now
32         ;
33 002372  113737 0000000 0000000 3$:   MOVB   CORUSR,CXBOWN ;;;Claim buffer for us
34 002400          2$:   ENABL          ;Enable interrupts
35         ;
36         ; Finished
37         ;
38 002406  000207          RETURN
39         .SBTTL  FRECXT -- Free the context block buffer
40         ;-----
41         ; This routine is called to release our ownership of the buffer
42         ; used to access the job context block of another job (CXTBUF).
43         ;
44 002410  123737 0000000 0000000 FRECXT: CMPB   CORUSR,CXBOWN ;Do we own the buffer now?
45 002416  001010          BNE    9$          ;Br if not
46 002420  105037 0000000          CLRB   CXBOWN        ;Say the buffer is free
47 002424  105037 0000000          CLRB   CXBJOB       ;Say no job's context data in buffer
48         ;
49         ; Restart any jobs waiting for the buffer
50         ;
51 002430  012700 0000000          MOV    #S$QCXB,RO    ;Get wait state
52 002434  004737 0000000          CALL   UREGO        ;Restart waiting jobs
53         ;
54         ; Finished
55         ;
56 002440  000207          9$:   RETURN

```

```

1          .SBTTL  GETCHA -- Get address of user's channel block
2          ;-----
3          ; This routine is called to obtain a pointer to a specified I/O channel
4          ; block for a specific job.
5          ; Note: GETCXT must have been called to gain exclusive access to the
6          ;       job context buffer before this routine can be used.
7          ;
8          ; Inputs:
9          ;   R2 = Job index number
10         ;   R3 = Channel number
11         ;
12         ; Outputs:
13         ;   R0 = Address of channel block
14         ;
15 002442 010346 GETCHA: MOV      R3, -(SP)
16         ;
17         ; Get the simulated RMON for the specified job into CXTBUF
18         ;
19 002444 004737 002510'      CALL    REDRMN      ;Get simulated RMON into CXTBUF
20         ;
21         ; Calculate address of specified channel block
22         ;
23 002450 013700 000000G      MOV      CXTBUF, R0      ;Point to RMON data in buffer
24 002454 062700 000000G      ADD      #R$CHN, R0      ;Point to 1st channel block
25 002460 020327 000021      CMP      R3, #17.      ;Is this channel in extended area?
26 002464 103404      BLO      1$      ;Br if not
27 002466 162703 000021      SUB      #17., R3      ;Get channel # within extended channel area
28 002472 062700 000000C      ADD      #R$XCHN-R$CHN, R0 ;Point to extended channel area
29 002476 070327 000000G  1$:    MUL      #CHNSIZ, R3      ;Multiply by # bytes per channel block
30 002502 060300      ADD      R3, R0      ;Get address of specified channel block
31         ;
32         ; Finished
33         ;
34 002504 012603      MOV      (SP)+, R3
35 002506 000207      RETURN

```

```
1 .SBTTL REDRMN -- Access simulated RMON for a job
2 ;-----
3 ; This routine is called to read into the job context block buffer
4 ; the simulated RMON (including I/O channel blocks) for a job.
5 ;
6 ; Inputs:
7 ; R2 = Index number of job whose RMON is to be read.
8 ;
9 ; Outputs:
10 ; (CXTBUF) = Rmon data.
11 ;
12 002510 010346 REDRMN: MOV R3, -(SP)
13 ;
14 ; Set up information about the simulated RMON
15 ;
16 002512 013703 0000000 MOV CXTRMN, R3 ;Get address of base of RMON with cxt blk
17 002516 012700 0000000 MOV #MVSIZ, R0 ;Get size of simulated RMON
18 ;
19 ; Get simulated RMON into context block buffer
20 ;
21 002522 004737 002726' CALL REDCXT ;Get RMON into buffer
22 ;
23 ; Finished
24 ;
25 002526 012603 MOV (SP)+, R3
26 002530 000207 RETURN
```

```

1          .SBTTL  GTUKBL -- Access region used for user key definitions
2          ;-----
3          ; This routine is used to access a 512-byte page of information in
4          ; the local named region used to store user key definitions.
5          ;
6          ; The form of the EMT is:
7          ;   .BYTE   0,126
8          ;   .WORD   17
9          ;   .WORD   job_number
10         ;   .WORD   block_number
11         ;   .WORD   destination_address
12         ;
13         ; The following error codes can be returned:
14         ;   0 = Job does not have any user-defined keys.
15         ;   1 = Job is swapped out of memory.
16         ;
17 002532  GTUKBL:
18         ;
19         ; Gain exclusive access to context buffer
20         ;
21 002532  004737  002324'      CALL   GETCXT      ;Gain access to context buffer
22 002536  103002             BCC    1$           ;Br if got it
23 002540  004737  000000G     CALL   STOP        ;Job was aborted while waiting
24         ;
25         ; Read KEYRCB cell into buffer
26         ;
27 002544  013702  000004G     1$:   MOV    EMTBLK+4,R2    ;Get number of job we are accessing
28 002550  012703  000000G     MOV    #KEYRCB,R3    ;Get address of cell we want
29 002554  012700  000002     MOV    #2,R0         ;Get # bytes we need
30 002560  004737  002726'     CALL   REDCXT       ;Copy data from jobs KEYRCB cell
31         ;
32         ; See if job has a region assigned for key definitions
33         ;
34 002564  013705  000000G     MOV    CXTBUF,R5    ;Get pointer to base of context buffer
35 002570  011503             MOV    (R5),R3      ;Get value of KEYRCB
36 002572  001005             BNE   2$           ;Br if job has some user-defined keys
37         ;
38         ; Job has no user-defined keys
39         ;
40 002574  004737  002410'     CALL   FRECTX       ;Free context block buffer
41 002600  005000             CLR   R0            ;Return error code 0
42 002602  000137  000000G     JMP   SETERR
43         ;
44         ; Job has some user-defined keys.
45         ; Get the region control block into context buffer.
46         ;
47 002606  012700  000000G     2$:   MOV    #RC$$SZ,R0 ;Get # bytes to access
48 002612  004737  002726'     CALL   REDCXT       ;Copy RCB into cxtbuf
49         ;
50         ; See if job is currently in memory.
51         ;
52 002616  032762  000000G 000000G  BIT    #INCOR,LSW(R2) ;Is job in memory now?
53 002624  001011             BNE   3$           ;Br if yes
54         ;
55         ; Job is currently swapped out of memory
56         ;
57 002626  016537  000000G 000000G  MOV    RC$BLK(R5),URO ;Return swap file block # to caller in R0

```

```

58 002634 004737 002410'          CALL  FRECXT          ;Free context block buffer
59 002640 012700 000001          MOV    #1,R0          ;Return error code 1
60 002644 000137 000000G        JMP    SETERR
61                               ;
62                               ; Job is in memory.
63                               ; Copy data from region into context buffer.
64                               ;
65 002650 016505 000000G        3#:   MOV    RC$BAS(R5),R5 ;Get 64-byte block # of phys base of region
66 002654 013703 000006G        MOV    EMTBLK+6,R3    ;Get desired page of region
67 002660 072327 000011        ASH   #9.,R3         ;Convert page # to byte offset
68 002664 012700 001000        MOV    #512.,R0     ;Say to move 512 bytes
69 002670 004737 000000G        CALL  CXBMOV        ;Copy data into CXTBUF
70                               ;
71                               ; Now move the data from the context buffer into the result buffer
72                               ;
73 002674 013705 000000G        MOV    CXTBUF,R5     ;Get current location of data
74 002700 013703 000010G        MOV    EMTBLK+10,R3 ;Get virtual address of result buffer
75 002704 012700 000400        MOV    #256.,R0     ;Get # words to move
76 002710 012546                4#:   MOV    (R5)+,-(SP) ;Get next word
77 002712 106623                MTPD  (R3)+         ;Move into result buffer
78 002714 077003                SOB   R0,4#         ;Loop till all moved
79                               ;
80                               ; Finished
81                               ;
82 002716 004737 002410'          CALL  FRECXT          ;Free context block buffer
83 002722 000137 000000G        JMP    EMTXIT

```

REDCXT -- Read a job context block into buffer

```

1          .SBTTL  REDCXT -- Read a job context block into buffer
2          ;-----
3          ; This routine is called to read a portion of a job context block,
4          ; belonging to some job other than the current job, into the
5          ; CXTBUF buffer.
6          ; GETCXT must have been called prior to calling this routine to
7          ; guarantee exclusive access to the context block buffer.
8          ;
9          ; Inputs:
10         ; R2 = Job index number of the job whose context block is to be read.
11         ; R3 = Address of item in context block which is to be positioned
12         ;       at the base of the buffer.
13         ; R0 = Number of bytes to be read (512 maximum).
14         ;
15         ; Outputs:
16         ; CXTBUF contains specified data.
17         ;
18 002726 010346 REDCXT: MOV     R3,-(SP)
19 002730 010546         MOV     R5,-(SP)
20         ;
21         ; See if the requested data is already in the buffer.
22         ;
23 002732 120237 000000G         CMPB   R2,CXBJOB      ;Is data for right job in buffer?
24 002736 001006         BNE    1$          ;Br if not
25 002740 020337 000000G         CMP    R3,CXBBAS      ;Is base item correct?
26 002744 001003         BNE    1$          ;Br if not
27 002746 023700 000000G         CMP    CXBSIZ,R0      ;Do we have enough data in buffer?
28 002752 103026         BHIS   9$          ;Br if yes -- We are finished
29         ;
30         ; We do not have the correct data in the buffer.
31         ; Set up information about what we will be getting into buffer.
32         ;
33 002754 110237 000000G 1$:   MOVB   R2,CXBJOB      ;Remember which context block we are getting
34 002760 010337 000000G         MOV    R3,CXBBAS      ;Remember base offset
35 002764 005200         INC    R0            ;Force size to be word multiple
36 002766 042700 000001         BIC    #1,R0
37 002772 010037 000000G         MOV    R0,CXBSIZ      ;Remember amt of data being gotten
38         ;
39         ; See if the job whose context data we want is currently in memory.
40         ;
41 002776 032762 000000G 000000G  BIT    ##INCOR,LSW(R2) ;Is job in memory now?
42 003004 001407         BEQ    2$          ;Br if not
43         ;
44         ; Job is in memory.
45         ; Do a memory-to-memory move to get the data.
46         ;
47 003006 016205 000000G         MOV    LCXPAR(R2),R5  ;Get base PAR mapping value
48 003012 162703 000000G         SUB    #CXTBAS,R3    ;Convert address to offset within cxt blk
49 003016 004737 000000G         CALL   CXBMOV        ;Move data to context block buffer
50 003022 000402         BR    9$
51         ;
52         ; Job is not now in memory.
53         ; Read the context block data from the swap file.
54         ;
55 003024 004737 003036' 2$:   CALL   CXBSWP        ;Read data from swap file
56         ;
57         ; Finished

```

```
58 ;  
59 003030 012605 9$: MOV (SP)+, R5  
60 003032 012603 MOV (SP)+, R3  
61 003034 000207 RETURN
```

```

1          .SBTTL  CXBSWP -- Read job context data from swap file
2          ;-----
3          ; Read some job context data from the swap file into CXTBUF buffer.
4          ;
5          ; Inputs:
6          ;   R2, CXBJOB = Index number of job whose context data is being read.
7          ;   R3, CXBBAS = Address within context block of item to be read into
8          ;                   the base of the context buffer.
9          ;   R0, CXBSIZ = Amt of data to be read (512. maximum)
10         ;
11 003036 010146 CXBSWP: MOV     R1, -(SP)
12 003040 010246      MOV     R2, -(SP)
13 003042 010346      MOV     R3, -(SP)
14 003044 010446      MOV     R4, -(SP)
15         ;
16         ; Determine which block in swap file has start of data being read
17         ;
18 003046 162703 0000000 SUB     #CXTBAS, R3      ; Convert address to offset within context
19 003052 010301      MOV     R3, R1          ; Get starting offset
20 003054 072127 177767 ASH     #-9, R1          ; Determine starting block # within context
21 003060 066201 0000000 ADD     LSWPBK(R2), R1   ; Add base block # within swap file
22         ;
23         ; Determine how much data to read this time
24         ;
25 003064 042703 177000 BIC     #^C<777>, R3    ; Get starting offset within 1st block
26 003070 010302      MOV     R3, R2
27 003072 060002      ADD     R0, R2          ; Get offset beyond last byte we need
28 003074 020227 001000 CMP     R2, #512.       ; We cannot read more than 512 bytes
29 003100 101402      BLOS   1$              ; Br if don't need to truncate
30 003102 012702 001000 MOV     #512., R2       ; Read 512 bytes this time
31         ;
32         ; Read data from swap file
33         ;
34 003106 006202 1$: ASR     R2              ; Convert # bytes to # words to read
35 003110      . READW  #USREMT, #USWPCH, CXTBUF, R2, R1
36         ;
37         ; If data is not positioned at the start of the block, move it
38         ; down to the base of the buffer.
39         ;
40 003144 005703      TST     R3              ; Is data at base of buffer now?
41 003146 001434      BEQ     9$              ; Br if yes -- Don't need to move
42 003150 006302      ASL     R2              ; Get back # bytes read
43 003152 160302      SUB     R3, R2          ; Get # bytes we need to move down
44 003154 010200      MOV     R2, R0          ; Get # bytes to move
45 003156 006200      ASR     R0              ; Get # words we need to move down
46 003160 013704 0000000 MOV     CXTBUF, R4      ; Point to base of buffer
47 003164 060403      ADD     R4, R3          ; Point to start of data we want
48 003166 012324 2$: MOV     (R3)+, (R4)+   ; Move the data down
49 003170 077002      SOB     R0, 2$        ; Loop till all moved
50         ;
51         ; See if the requested data spans a block boundary.
52         ;
53 003172 013703 0000000 MOV     CXBSIZ, R3      ; Get total amt of data requested
54 003176 160203      SUB     R2, R3          ; Get # left to be read
55 003200 003417      BLE     9$              ; Br if we have gotten all that was requested
56         ;
57         ; Read in second block to get remainder of data.

```

```
58 ; Data from second block is read on top of data from first block
59 ; so that we don't have to move it.
60 ;
61 003202 005201 ; INC R1 ; Increment block number
62 003204 006203 ; ASR R3 ; Get # words to read
63 003206 ; . READW #USREMT, #USWPCH, R4, R3, R1
64 ;
65 ; Finished
66 ;
67 003240 012604 9$: MOV (SP)+, R4
68 003242 012603 MOV (SP)+, R3
69 003244 012602 MOV (SP)+, R2
70 003246 012601 MOV (SP)+, R1
71 003250 000207 RETURN
```

```

1          .SBTTL .CSISPC & .CSIGEN
2          ;-----
3          ; Command string interpreters.
4          ;
5 003252 112737 000001 000000G CSISPC: MOVB #1,SPCFLG ;REMEMBER THIS IS .CSISPC
6 003260 000420          BR CSICOM
7 003262 105037 000000G CSIGEN: CLRB SPCFLG ;REMEMBER THIS IS .CSIGEN
8          ;
9          ; Close user's channels 0-10
10         ;
11 003266 012705 000010          MOV #10,R5 ;GET LAST CHANNEL # TO CLOSE
12 003272          1$: .CLOSE R5 ;CLOSE THE CHANNEL
13 003302 005305          DEC R5 ;MORE TO DO?
14 003304 002372          BGE 1$ ;BR IF YES
15         ; .CSIGEN returns address above user's handler space in R0.
16 003306 013737 000010G 000000G MOV EMTBLK+10,UR0 ;RETURN HANDLER SPACE ADDRESS IN R0
17 003314 042737 000001 000000G BIC #1,UR0 ;MAKE SURE ADDRESS IS EVEN
18         ;
19         ; Save the user's stack pointer on entry in case we have to
20         ; restore it because we have a command error.
21         ;
22 003322 106506 CSICOM: MFPD SP ;Get user's stack pointer
23 003324 012637 000000G MOV (SP)+,CSIUSP ;Save in job context block cell
24         ;
25         ; Print prompt and get command line
26         ; or move command from user's area to our area.
27         ;
28 003330 013703 000004G CSIGCL: MOV EMTBLK+4,R3 ;ADDRESS OF USER'S COMMAND STRING
29 003334 001417          BEQ 2$ ;BR IF INPUT IS FROM THE TERMINAL
30         ; Move command string from user's area to our internal buffer.
31 003336 010300          MOV R3,R0 ;CHECK IF ADDRESS IS LEGAL
32 003340 004737 000000G          CALL VALADB
33 003344 012702 000000G          MOV #CSIBUF,R2 ;PUT COMMAND HERE
34 003350 004737 000000G          1$: CALL GETUCH ;GET CHAR FROM USER'S AREA
35 003354 020227 000120G          CMP R2,#<CSIBUF+80.>;DON'T GO BEYOND END OF BUFFER
36 003360 103003          BHI 9$ ;BR IF COMMAND LINE TOO LONG
37 003362 110022          MOVB R0,(R2)+ ;MOVE TO OUR BUFFER
38 003364 001371          BNE 1$ ;LOOP TILL ASCIZ NULL HIT
39 003366 000435          BR 3$ ;GO PROCESS THE COMMAND
40 003370 105012          9$: CLRB (R2) ;STORE NULL AT END OF COMMAND
41 003372 000433          BR 3$ ;GO PROCESS THE COMMAND
42         ; Prompt for and accept a command from the terminal.
43 003374 005005          2$: CLR R5 ;ASSUME WE DON'T NEED TO SUSPEND COMMAND FILES
44 003376 023727 000010G 000003G CMP EMTBLK+10,#3 ;IS THIS A .GTLIN EMT?
45 003404 101010          BHI 6$ ;BR IF NOT
46 003406 001004          BNE 8$ ;BR IF NOT FORCING INPUT FROM TERMINAL
47 003410 013705 000000G          MOV CFPNT,R5 ;SAVE CURRENT COMMAND FILE BUFFER POINTER
48 003414 005037 000000G          CLR CFPNT ;SUSPEND INPUT FROM COMMAND FILE
49 003420 013702 000006G          8$: MOV EMTBLK+6,R2 ;GET ADDRESS OF USER'S PROMPT STRING
50 003424 000405          BR 7$
51 003426 012702 000000G          6$: MOV #USRUCA,R2 ;PUT * IN USER'S AREA FOR PROMPT
52 003432 012746 100052          MOV #100052,-(SP)
53 003436 106612          MTPD (R2) ;MOVE TO USER'S AREA
54 003440 012703 000000G          7$: MOV #CSIBUF,R3 ;PUT STRING HERE
55 003444          DCALL ASKLIN ;PRINT PROMPT & ACCEPT LINE FROM TERMINAL
56 003452 005705          TST R5 ;DID WE SUSPEND INPUT FROM COMMAND FILES?
57 003454 001402          BEQ 3$ ;BR IF NOT
    
```

```

58 003456 010537 0000000      MOV      R5,CFPNT      ;REENABLE INPUT FROM COMMAND FILE
59
60      ; The command line is now stored in CSIBUF in asciz form.
61      ; Copy the string to user's buffer if requested.
62
63 003462 032737 000001 0000100 3$:      BIT      #1,EMTBLK+10  ;DOES USER WANT A COPY OF COMMAND LINE?
64 003470 001424      BEQ      CSISCN      ;BR IF NOT
65      ; Move command line to user's buffer.
66 003472 013703 0000120      MOV      EMTBLK+12,R3  ;GET ADDRESS OF BUFFER WHERE COMMAND GOES
67 003476 010300      MOV      R3,R0
68 003500 004737 0000000      CALL     VALADB      ;MAKE SURE ADDRESS IS LEGAL
69 003504 012702 0000000      MOV      #CSIBUF,R2  ;POINT TO BUFFER HOLDING COMMAND
70 003510 112200      5$:      MOVB     (R2)+,R0  ;GET CHAR FROM COMMAND LINE
71 003512 001403      BEQ      4$          ;BR WHEN ASCIZ NULL HIT
72 003514 004737 0000000      CALL     PUTUCH     ;MOVE CHAR TO USER'S BUFFER
73 003520 000773      BR      5$
74 003522 004737 0000000      4$:      CALL     PUTUCH     ;MOVE TERMINATING NULL
75
76      ; If this is .GTLIN EMT, we are finished
77
78 003526 023727 0000100 0000003      CMP      EMTBLK+10,#3  ;IS THIS A .GTLIN EMT?
79 003534 101002      BHI     CSISCN      ;BR IF NOT
80 003536 000137 0000000      JMP      EMTXIT     ;WE ARE FINISHED WITH .GTLIN
    
```

```

1      ;
2      ; At this point the command string has been accrued and is stored
3      ; in asciz form in CSIBUF.
4      ;
5      ; If we are doing a .CSISPC, zero user's file spec block.
6 003542 105737 000000G CSISCN: TSTB   SPCFLG           ; DOING .CSISPC?
7 003546 001417          BEQ     4$              ; BR IF NOT
8 003550 013705 000010G   MOV    EMTBLK+10,R5       ; GET ADDRESS OF FILE SPEC BUFFER
9 003554 042705 000001   BIC    #1,R5              ; MAKE ADDRESS EVEN
10 003560 010500         MOV    R5,R0              ; MAKE SURE ADDRESS IS LEGAL
11 003562 062700 000116   ADD    #2*39.,R0
12 003566 004737 000000G   CALL  VALADW
13 003572 010504         MOV    R5,R4
14 003574 012703 000047   MOV    #39.,R3           ; FILE SPEC IS 39 WORDS LONG
15 003600 005046 1$:    CLR    -(SP)           ; ZERO USER'S AREA
16 003602 106624         MTPD   (R4)+
17 003604 077303         SOB    R3,1$
18      ;
19      ; The following registers are set up for the actual command scan:
20      ; R1 = Pointer into CSIBUF.
21      ; R4 = File number we are working on.
22      ; R5 = Pointer into user's device spec buffer (.CSISPC only).
23      ;
24      ; If command line contains an equal sign then we have output file specs.
25      ; Otherwise first file is an input file (# 3).
26      ;
27 003606 105037 000000G 4$:    CLRB   CSIEQL           ; HAVEN'T SEEN EQUAL SIGN YET
28 003612 105037 000000G   CLRB   SWTCNT           ; NO SWITCHES YET
29 003616 013737 000002' 000000G   MOV    R5ODK,CSIDEV     ; SET DEFAULT DEVICE NAME
30 003624 012701 000000G   MOV    #CSIBUF,R1       ; POINT TO COMMAND STRING
31 003630 010102         MOV    R1,R2
32 003632 112200 2$:    MOVB   (R2)+,R0       ; SCAN ACROSS COMMAND LOOKING FOR EQUAL SIGN
33 003634 001410         BEQ    3$              ; BR IF END OF COMMAND HIT -- NO EQUAL SIGN FOUND
34 003636 120027 000075   CMPB   R0,#'=          ; DOES COMMAND CONTAIN AN EQUAL SIGN?
35 003642 001403         BEQ    5$              ; BR IF YES
36 003644 120027 000074   CMPB   R0,#'<        ; LESS-THAN SIGN IS EQUIVALENT TO EQUAL-SIGN
37 003650 001370         BNE    2$
38      ; Command line contains an equal sign. Output files come first.
39 003652 005004 5$:    CLR    R4              ; FIRST FILE # IS 0
40 003654 000406         BR     CSICHR
41      ; Command line does not contain an equal sign. Output files are omitted.
42 003656 012704 000003 3$:    MOV    #3,R4           ; FIRST FILE IS # 3
43 003662 062705 000036   ADD    #36,R5           ; POINT TO 1ST INPUT FILE SPEC AREA
44 003666 105237 000000G   INCB   CSIEQL           ; SAY WE ARE BEYOND EQUAL SIGN

```

```

1
2 ; Begin processing the next file specification.
3
4 003672 004737 004660' CSICHR: CALL CSIGET ; GET NEXT CHAR FROM COMMAND LINE
5 003676 105700 CSICH1: TSTB R0 ; REACHED END OF COMMAND LINE?
6 003700 001002 BNE 7$ ; BR IF NOT
7 003702 000137 004504' JMP CSIFIN ; FINISH OFF COMMAND
8 003706 120027 000075 7$: CMPB R0,#'=' ; REACHED EQUAL SIGN?
9 003712 001403 BEQ 17$ ; BR IF YES
10 003714 120027 000074 CMPB R0,#'< ; LESS-THAN SIGN IS EQUIVALENT TO EQUAL SIGN
11 003720 001023 BNE 1$ ; BR IF NOT
12 ; "=" means start of input file specs.
13 003722 105737 000000G 17$: TSTB CSIEQL ; HAVE WE SEEN AN EQUAL SIGN BEFORE?
14 003726 001402 BEQ 11$ ; BR IF NOT
15 003730 000137 004520' JMP CSIILL ; ERROR IF 2 EQUAL SIGNS
16 003734 105237 000000G 11$: INCB CSIEQL ; REMEMBER EQUAL SIGN SEEN
17 003740 012704 000003 MOV #3,R4 ; NEXT FILE # IS 3
18 003744 013705 000010G MOV EMTBLK+10,R5 ; GET ADDRESS OF USER'S FILE SPEC AREA
19 003750 042705 000001 BIC #1,R5 ; MAKE EVEN
20 003754 062705 000036 ADD #36,R5 ; POINT TO SPACE FOR 1ST INPUT FILE SPEC
21 003760 013737 000002' 000000G MOV R50DK,CSIDEV ; RESET DEFAULT DEVICE TO "DK:"
22 003766 000741 BR CSICHR ; CONTINUE SCAN
23 ; See if we have a null file spec.
24 003770 120027 000054 1$: CMPB R0,#' ; NULL FILE SPEC?
25 003774 001002 BNE 14$ ; BR IF NOT
26 003776 000137 004460' JMP CSINXT ; ADVANCE TO NEXT FILE
27 ; See if we have a leading switch specification.
28 004002 120027 000057 14$: CMPB R0,#'/' ; START OF SWITCH SPECIFICATION?
29 004006 001002 BNE 15$ ; BR IF NOT
30 004010 000137 004336' JMP CSISWT ; GO PROCESS THE SWITCH
31
32 ; Accrue a file specification.
33
34 004014 005301 15$: DEC R1 ; POINT TO 1ST CHARACTER OF NAME
35 004016 020427 000010 CMP R4,#8. ; HAVE WE ALREADY GOTTEN MAX # FILES ALLOWED?
36 004022 101006 BHI 13$ ; BR IF TOO MANY
37 004024 105737 000000G TSTB CSIEQL ; ARE WE GETTING INPUT OR OUTPUT FILES?
38 004030 001005 BNE 2$ ; BR IF INPUT FILES
39 004032 020427 000003 CMP R4,#3 ; ONLY ALLOW 3 OUTPUT FILES
40 004036 103402 BLD 2$ ; BR IF OK
41 004040 000137 004520' 13$: JMP CSIILL ; ERROR
42 ; Get the default extension for the file name.
43 004044 010400 2$: MOV R4,R0 ; GET FILE NUMBER
44 004046 020027 000002 CMP R0,#2 ; INPUT OR OUTPUT FILE?
45 004052 101402 BLOS 12$ ; BR IF OUTPUT FILE
46 004054 005000 CLR R0 ; USE EXTENSION 0 FOR ALL INPUT FILES
47 004056 000402 BR 16$
48 004060 005200 12$: INC R0 ; CONVERT FILE # TO INDEX INTO EXTENSION TABLE
49 004062 006300 ASL R0 ; 2 BYTES PER ENTRY
50 004064 063700 000006G 16$: ADD EMTBLK+6,R0 ; ADD BASE OF DEFAULT EXT TABLE
51 004070 004737 000000G CALL VALADW ; MAKE SURE ADDRESS IS LEGAL
52 004074 106510 MFPD (R0) ; GET DEFAULT EXTENSION FROM USER'S TABLE
53 004076 012600 MOV (SP)+,R0 ; GET DEFAULT EXTENSION
54 ; Parse the device-file specification.
55 004100 004737 004726' CALL ACRFIL ; ACCRUE THE FILE SPECIFICATION
56 004104 103002 BCC 19$ ; BR IF NO ERROR
57 004106 000137 004520' JMP CSIILL ; INVALID SPEC

```

.CSISPC &amp; .CSIGEN

```

58 ; Require a file name with file structured output devices.
59 004112 020427 000003 19#: CMP R4,#3 ; IS THIS AN OUTPUT FILE SPEC?
60 004116 103037 BHS 4# ; BR IF NOT
61 004120 .DSTATUS #CSIARE,#CSIFIL; GET INFO ABOUT THE DEVICE
62 004132 103020 BCC 9# ; BR IF VALID DEVICE
63 004134 105737 000000G TSTB SPCFLG ; IS THIS .CSISPC OR .CSIGEN?
64 004140 001571 BEQ ILLDEV ; BR IF .CSIGEN WITH INVALID DEVICE
65 ;
66 ; .CSISPC: Move file spec to user's area.
67 ;
68 004142 012700 000004 21#: MOV #4,R0 ; ASSUME 4 WORDS TO MOVE
69 004146 020427 000003 CMP R4,#3 ; INPUT OR OUTPUT FILE SPEC?
70 004152 103001 BHS 5# ; BR IF INPUT
71 004154 005200 INC R0 ; MOVE 5 WORDS FOR OUTPUT FILE SPECS
72 004156 012702 000000G 5#: MOV #CSIFIL,R2 ; POINT TO BUFFER WITH FILE SPEC WE GOT
73 004162 010503 MOV R5,R3 ; GET POINTER TO FILE SPEC AREA
74 004164 012246 6#: MOV (R2)+,-(SP) ; MOVE A WORD FROM FILE SPEC TO USER'S AREA
75 004166 106623 MTPD (R3)+
76 004170 077003 SOB R0,6# ; MOVE ALL OF SPEC
77 004172 000457 BR 10#
78 ;
79 ; .CSIGEN: See if file specified or device non-file structured.
80 ;
81 004174 105737 000000G 9#: TSTB SPCFLG ; .CSISPC?
82 004200 001360 BNE 21# ; IF SO, DON'T REQUIRE FILE NAME
83 004202 005737 000002G TST CSIFIL+2 ; WAS A FILE NAME SPECIFIED?
84 004206 001003 BNE 4# ; BR IF YES
85 004210 005737 000000G TST CSIARE ; IS DEVICE FILE STRUCTURED?
86 004214 002541 BLT CSIILL ; ERROR IF YES
87 ;
88 ; .CSIGEN: Do .lookup or .enter for file.
89 ;
90 004216 105737 000000G 4#: TSTB SPCFLG ; CHECK FOR .CSISPC OR .CSIGEN
91 004222 001347 BNE 21# ; BR IF .CSISPC
92 004224 020427 000003 CMP R4,#3 ; INPUT OR OUTPUT FILE?
93 004230 103025 BHS 8# ; BR IF INPUT
94 ; Output: Enter the file.
95 004232 .ENTER #CSIARE,R4,#CSIFIL,CSIFIL+10
96 004262 103023 BCC 10# ; BR IF ENTER WAS SUCCESSFUL
97 004264 123727 000000G 000003 CMPB @#ERRLOC,#3 ; PROTECTED FILE CONFLICT?
98 004272 001002 BNE 18# ; BR IF NOT
99 004274 000137 004532' JMP CSIPRO ; RETURN ERROR CODE 2
100 004300 000137 004540' 18#: JMP CSIFUL ; RETURN ERROR CODE 3
101 ; Input: Lookup the file.
102 004304 8#: .LOOKUP #CSIARE,R4,#CSIFIL
103 004326 103001 BCC 10# ; BR IF OK
104 004330 000506 BR CSINGF ; NO FILE FOUND
105 004332 000137 003672' 10#: JMP CSICHR ; CONTINUE SCAN

```

```

1
2 ; Process a switch specification.
3
4 004336 004737 004660' CSISWT: CALL CSIGET ;GET NEXT CHAR
5 004342 010003 MOV R0,R3 ;GET SWITCH CHARACTER
6 004344 001465 BEQ CSIILL ;BR IF NO CHAR FOLLOWS "/"
7 004346 042703 177400 3$: BIC #^C<377>,R3 ;LEAVE ONLY 8-BITS
8 004352 010400 MOV R4,R0 ;GET FILE #
9 004354 000300 SWAB R0 ;PUT IN LEFT BYTE
10 004356 050003 BIS R0,R3 ;COMBINE FILE # WITH SWITCH CHARACTER
11 ; See if switch has an associated value.
12 004360 004737 004660' CALL CSIGET ;GET NEXT CHAR
13 004364 120027 000072 CMPB R0,#'; ;IS THERE AN ASSOCIATED VALUE?
14 004370 001023 BNE 2$ ;BR IF NOT
15
16 ; Switch has a value -- Get it and push onto user's stack.
17
18 004372 052703 100000 BIS #100000,R3 ;SET VALUE FLAG
19 004376 004737 005324' 1$: CALL SWVAL ;GET VALUE
20 004402 103470 BCS BADVAL ;BR IF INVALID
21 ; Push switch value onto user's stack.
22 004404 004737 005704' CALL UPUSH ;PUSH ONTO USER'S STACK
23 004410 105237 000000G INCB SWTCNT ;COUNT # SWITCHES ACCRUED
24 ; Push switch control word on user's stack.
25 004414 010300 MOV R3,R0 ;GET CONTROL WORD
26 004416 004737 005704' CALL UPUSH ;PUSH ONTO USER'S STACK
27 ; See if there are additional values for this switch.
28 004422 004737 004660' CALL CSIGET ;GET NEXT CHAR
29 004426 120027 000072 CMPB R0,#'; ;IS THERE ANOTHER VALUE
30 004432 001761 BEQ 1$ ;BR IF YES
31 004434 000137 003676' JMP CSICHI ;CONTINUE SCAN
32
33 ; No switch value
34
35 004440 005301 2$: DEC R1 ;BACK UP CHARACTER POINTER
36 004442 010300 MOV R3,R0 ;GET SWITCH CONTROL WORD
37 004444 004737 005704' CALL UPUSH ;PUSH ONTO USER'S STACK
38 004450 105237 000000G INCB SWTCNT ;COUNT # OF SWITCHES
39 004454 000137 003672' JMP CSICHR ;CONTINUE SCAN
40
41 ; Finished a file spec.
42
43 004460 062705 000010 CSINXT: ADD #10,R5 ;POINT TO NEXT FILE SPEC AREA
44 004464 020427 000003 CMP R4,#3 ;DOING AN INPUT OR OUTPUT FILE?
45 004470 103002 BHS 1$ ;BR IF AN INPUT FILE
46 004472 062705 000002 ADD #2,R5 ;SKIP 2 MORE BYTES
47 004476 005204 1$: INC R4 ;ADVANCE FILE #
48 004500 000137 003672' JMP CSICHR ;CONTINUE SCAN
49
50 ; Finished scanning command line.
51 ; Return number of switches on top of user's stack.
52
53 004504 113700 000000G CSIFIN: MOVB SWTCNT,R0 ;GET # SWITCHES ACCRUED
54 004510 004737 005704' CALL UPUSH ;PUSH ONTO USER'S STACK
55 004514 000137 000000G JMP EMTXIT ;FINISHED

```

```

1          ;
2          ; CSI errors.
3          ;
4          ; Illegal command
5 004520 005003 CSIILL: CLR    R3          ;ERROR CODE = 0
6 004522 000424          BR      CSIERR
7          ; Illegal device
8 004524 012703 000001 ILLDEV: MOV    #1,R3          ;ERROR CODE = 1
9 004530 000421          BR      CSIERR
10         ; Enter failed -- Protected file conflict.
11 004532 012703 000002 CSIPRO: MOV    #2,R3          ;ERROR CODE = 2
12 004536 000416          BR      CSIERR
13         ; Enter failed -- Insufficient disk space.
14 004540 012703 000003 CSIFUL: MOV    #3,R3          ;ERROR CODE = 3
15 004544 000413          BR      CSIERR
16         ; Lookup failed
17 004546 012703 000004 CSINOF: MOV    #4,R3          ;ERROR CODE = 4
18 004552 000410          BR      CSIERR
19         ; Invalid switch
20 004554 005003 BADSWT: CLR    R3          ;ERROR CODE = 0
21 004556 012702 000005          MOV    #5,R2          ;SET ERROR MESSAGE NUMBER
22 004562 000405          BR      CSIER1
23         ; Invalid switch value
24 004564 005003 BADVAL: CLR    R3          ;ERROR CODE = 0
25 004566 012702 000006          MOV    #6,R2          ;ERROR MESSAGE NUMBER
26 004572 000401          BR      CSIER1
27         ;
28         ; Common CSI error processing.
29         ;
30 004574 010302 CSIERR: MOV    R3,R2          ;SET ERROR MESSAGE NUMBER
31         ; Restore initial user stack pointer
32 004576 013746 000000G CSIER1: MOV    CSIUSP,-(SP)  ;Get initial stack pointer for user
33 004602 106606          MTPD   SP          ;Reset for user
34         ; If input is coming from the terminal, print error message and
35         ; retry command.
36 004604 005737 000004G          TST    EMTBLK+4          ;INPUT FROM THE TERMINAL?
37 004610 001020          BNE    3$          ;BR IF NOT
38 004612          DCALL  CSIMSG          ;PRINT ERROR MESSAGE
39         ; If this is .CSIGEN, purge channels we opened.
40 004620 105737 000000G          TSTB   SPCFLG          ;CSISPC OR CSIGEN?
41 004624 001010          BNE    1$          ;BR IF CSISPC
42 004626 005704          TST    R4          ;DID WE OPEN ANY CHANNELS?
43 004630 001406          BEQ    1$          ;BR IF NOT
44 004632          2$: .PURGE R4          ;PURGE A CHANNEL
45 004642 005304          DEC    R4          ;MORE TO DO?
46 004644 002372          BGE    2$          ;BR IF YES
47 004646 000137 003330'          1$: JMP    CSIGCL          ;GO BACK AND RETRY COMMAND
48         ; Input is coming from a user supplied string. Return an error code.
49 004652 010300          3$: MOV    R3,R0          ;GET ERROR CODE
50 004654 000137 000000G          JMP    SETERR          ;ABORT EMT

```

\*\*\* Subroutines \*\*\*

```

1          .SBTTL *** Subroutines ***
2          .SBTTL CSIGET -- Get next character from CSI command line
3          ;-----
4          ; CSIGET is called to get the next non-blank character from the CSI
5          ; command line. Lower case letters are converted to upper case.
6          ;
7          ; Inputs:
8          ; R1 = Pointer to character to be gotten
9          ;
10         ; Outputs:
11         ; R0 = Character acquired
12         ; R1 = Pointer past character gotten
13         ; z-flag set if character = null
14         ;
15 004660 112100 CSIGET: MOVB (R1)+,R0      ;GET NEXT CHARACTER
16 004662 001420      BEQ 2$              ;BR IF HIT STRING DELIMITER
17 004664 120027 000040      CMPB RO,#'      ;IS CHAR A SPACE?
18 004670 001773      BEQ CSIGET          ;SKIP OVER SPACES
19 004672 120027 0000006      CMPB RO,#TAB    ;IS CHARACTER A TAB?
20 004676 001770      BEQ CSIGET          ;SKIP TABS
21 004700 120027 000141      CMPB RO,#141   ;LOWER CASE A?
22 004704 103405      BLD 1$              ;BR IF NOT LOWER CASE LETTER
23 004706 120027 000172      CMPB RO,#172   ;LOWER CASE Z?
24 004712 101002      BHI 1$              ;BR IF NOT LOWER CASE LETTER
25 004714 162700 000040      SUB #40,R0     ;CONVERT LOWER CASE TO UPPER CASE
26 004720 042700 177600      1$: BIC #^C<177>,R0 ;RETURN ONLY 7-BIT CHAR
27 004724 000207      2$: RETURN

```

ACRFIL -- Accrue a file specification

```

1          .SBTTL  ACRFIL -- Accrue a file specification
2
3          ;-----
4          ; ACRFIL is called to parse a file specification of the form
5          ; dev:file.ext[size].
6
7          ; Inputs:
8          ; R0 = Default extension
9          ; R1 = Pointer to start of file spec
10         ; CSIDEV = Default device (Rad50)
11
12         ; Outputs:
13         ; R1 = Pointer to character past end of file spec
14         ; CSIFIL = 5 word buffer containing Rad50 file spec accrued
15         ; CSIDEV is updated if a device is explicitly specified
16         ; C-flag is set on return if the file spec is invalid
17 004726 010446 ACRFIL: MOV     R4,-(SP)
18 004730 010546         MOV     R5,-(SP)
19
20         ; Initialize CSIFIL buffer.
21
22 004732 012705 00000000         MOV     #CSIFIL,R5         ;POINT TO RESULT BUFFER
23 004736 013725 00000000         MOV     CSIDEV,(R5)+         ;PUT IN DEFAULT DEVICE
24 004742 005025 1#: CLR      (R5)+         ;CLEAR REST OF BUFFER
25 004744 020527 00001000         CMP     R5,#CSIFIL+0.
26 004750 101774         BLOS   1#
27 004752 010037 00000060         MOV     R0,CSIFIL+6         ;SET DEFAULT EXTENSION
28
29         ; First character of file spec must be valid Rad50 character or period.
30
31 004756 004737 004660'         CALL    CSIGET         ;GET 1ST CHAR OF FILE SPEC
32 004762 005301         DEC     R1         ;BACKUP CHAR POINTER SO WE WILL GET CHAR AGAIN
33 004764 120027 0000056         CMPB   R0,#'.'         ;PERIOD?
34 004770 001403         BEQ    6#         ;BR IF YES
35 004772 004737 005222'         CALL    R50CHR         ;VALID RAD50 CHARACTER?
36 004776 103453         BCS    10#         ;BR IF INVALID
37
38         ; Get first field which will be device or file name.
39
40 005000 004737 005136' 6#: CALL    GTRD50         ;ACCRUE RAD50 VALUE
41 005004 121127 0000072         CMPB   (R1),#':         ;IS THIS A DEVICE OR FILE NAME?
42 005010 001012         BNE    2#         ;BR IF FILE NAME
43 005012 005201         INC     R1         ;POINT PAST COLON
44 005014 010437 00000000         MOV     R4,CSIFIL         ;SET DEVICE NAME
45 005020 001442         BEQ    10#         ;BR IF NULL DEVICE NAME
46 005022 010437 00000000         MOV     R4,CSIDEV         ;SET NEW DEFAULT DEVICE NAME
47 005026 005705         TST    R5         ;WAS DEVICE NAME LONGER THAN 3 CHARS?
48 005030 001036         BNE    10#         ;ERROR IF YES
49
50         ; Accrue the file name.
51
52 005032 004737 005136'         CALL    GTRD50         ;ACCRUE NEXT RAD50 FIELD
53 005036 010437 0000020 2#: MOV     R4,CSIFIL+2         ;FIRST 3 CHARS OF FILE NAME
54 005042 010537 0000040         MOV     R5,CSIFIL+4         ;SECOND 3 CHARS OF FILE NAME
55 005046 121127 0000056         CMPB   (R1),#'.         ;IS AN EXTENSION SPECIFIED?
56 005052 001010         BNE    3#         ;BR IF NOT
57 005054 005201         INC     R1         ;POINT PAST PERIOD

```

ACRFIL -- Accrue a file specification

```

58 ;
59 ; Accrue the extension.
60 ;
61 005056 004737 005136'      CALL   GTRD50      ; ACCRUE THE EXTENSION FIELD
62 005062 010437 000006G      MOV    R4,CSIFIL+6 ; STORE THE EXTENSION
63 005066 121127 000056      CMPB   (R1),#'     ; Was another period specified?
64 005072 001415              BEQ    10$         ; Br if yes -- Error
65 ;
66 ; See if file size was specified.
67 ;
68 005074 121127 000133      3$:   CMPB   (R1),#'E      ; WAS FILE SIZE SPECIFIED?
69 005100 001010              BNE    4$         ; BR IF NOT
70 005102 005201              INC    R1         ; POINT PAST OPEN BRACKET
71 ; Get the file size
72 005104 004737 005476'      CALL   ACRDEC      ; ACCRUE A DECIMAL VALUE
73 005110 010037 000010G      MOV    R0,CSIFIL+8. ; STORE FILE SIZE
74 005114 122127 000135      CMPB   (R1)+, #'J    ; GOT PROPER DELIMITER?
75 005120 001002              BNE    10$         ; ERROR IF NOT
76 ;
77 ; Successful return
78 ;
79 005122 000241              4$:   CLC                    ; SIGNAL GOOD RETURN
80 005124 000401              BR     5$
81 ;
82 ; Error return
83 ;
84 005126 000261              10$:  SEC                    ; SIGNAL ERROR RETURN
85 005130 012605              5$:   MOV    (SP)+,R5
86 005132 012604              MOV    (SP)+,R4
87 005134 000207              RETURN

```

```

1          .SBTTL  GTRD50 -- Accrue a RAD50 name
2          ;-----
3          ; GTRD50 is called to accrue a name and return it as a Rad50 value.
4          ;
5          ; Inputs:
6          ; R1 = Pointer to start of name
7          ; SPCFLG: 1==> "*" and "%" legal in name; 0==> not legal.
8          ;
9          ; Outputs
10         ; R1 = Pointer left pointing to name delimiter
11         ; R4 = First 3 characters of name in Rad50
12         ; R5 = Second 3 characters of name
13         ;
14 005136 012746 022000 GTRD50: MOV #22000, -(SP) ; COUNTER TO CONTROL # CHARS ACQUIRED
15 005142 005005 4#: CLR R5 ; ACCRUE VALUE IN R5
16         ; Get next character and see if it is valid for Rad50 name.
17 005144 004737 004660' 1#: CALL CSIGET ; GET NEXT CHARACTER
18 005150 004737 005222' CALL R5OCHR ; IS THIS A VALID RAD50 CHAR?
19 005154 103001 BCC 2# ; BR IF VALID
20 005156 005301 DEC R1 ; BACK UP CHAR POINTER TO DELIMITER
21         ; We got a valid Rad50 character.
22         ; Add to accumulated value.
23 005160 070527 000050 2#: MUL #50, R5 ; MULTIPLY BY 50
24 005164 060005 ADD R0, R5 ; ADD IN NEW CHARACTER
25 005166 006316 ASL (SP) ; HAVE WE GOTTEN 3 CHARS YET?
26 005170 103365 BCC 1# ; BR IF NOT
27 005172 005716 TST (SP) ; HAVE WE GOTTEN 6 CHARS?
28 005174 001402 BEQ 3# ; BR IF YES
29 005176 010504 MOV R5, R4 ; RETURN 1ST 3 CHARS IN R4
30 005200 000760 BR 4# ; GO GET NEXT 3 CHARS
31         ; Skip any chars in name after sixth.
32 005202 004737 004660' 3#: CALL CSIGET ; GET NEXT CHAR
33 005206 004737 005222' CALL R5OCHR ; IS THIS A DELIMITER?
34 005212 103373 BCC 3# ; SKIP IT IF NOT
35 005214 005301 DEC R1 ; POINT BACK TO DELIMITER
36 005216 005726 TST (SP)+ ; REMOVE COUNT FROM STACK
37 005220 000207 RETURN

```

```

1          .SBTTL  R50CHR -- Convert ascii character to Rad50 value
2          ;-----
3          ; R50CHR is called to convert an ascii character to its equivalent
4          ; Rad50 value.
5          ;
6          ; Inputs
7          ; RO = Ascii character
8          ; SPCFLG: 1==> allow "*" and "%"; 0==> do not allow * and %
9          ;
10         ; Outputs
11         ; RO = Rad50 value of character
12         ; C-flag set if character is not a valid Rad50 character (value returned = 0)
13         ;
14 005222 010146 R50CHR: MOV     R1, -(SP)
15 005224 012701 005302'   MOV     #R50RNG, R1      ; POINT TO CONVERSION TABLE
16         ; Search for ascii char in our conversion table
17 005230 120011 1$:     CMPB   RO, (R1)      ; COMPARE WITH LOW VALUE IN RANGE
18 005232 103413         BLO     2$              ; BR IF TOO SMALL FOR RANGE
19 005234 120061 000001   CMPB   RO, 1(R1)      ; COMPARE WITH HIGH VALUE IN RANGE
20 005240 101010         BHI     2$              ; BR IF TOO LARGE FOR RANGE
21 005242 122121         CMPB   (R1)+, (R1)+    ; IS THIS A SPECIAL CHAR (* OR %)?
22 005244 001003         BNE     3$              ; BR IF NOT
23 005246 105737 000000G  TSTB   SPCFLG        ; SHOULD WE ALLOW SPECIAL CHARS?
24 005252 001407         BEQ     9$              ; BR IF NOT
25         ; We have a valid character. Convert ascii value to rad50.
26 005254 061100 3$:     ADD     (R1), RO      ; CONVERT ASCII VALUE TO RAD50 VALUE
27 005256 000241         CLC                    ; SIGNAL GOOD RETURN
28 005260 000406         BR      10$             ; GO RETURN
29         ; Try next range in table.
30 005262 062701 000004 2$:     ADD     #4, R1      ; POINT TO NEXT ENTRY IN CONVERSION TABLE
31 005266 105711         TSTB   (R1)          ; HAVE WE HIT END OF TABLE?
32 005270 001357         BNE     1$              ; BR IF NOT
33         ; Invalid character
34 005272 005000 9$:     CLR     RO              ; RETURN 0 AS VALUE
35 005274 000261         SEC                    ; SIGNAL ERROR
36         ; Return
37 005276 012601 10$:    MOV     (SP)+, R1
38 005300 000207         RETURN
39         ;
40         ; Rad50 conversion table.
41         ; Values: Low-value, high-value, Rad50-Ascii-value
42         ;
43 005302      101      132 R50RNG: .BYTE  'A, 'Z      ; A-Z
44 005304 177700         .WORD  1-'A
45 005306      060      071 .BYTE  '0, '9      ; 0-9
46 005310 177756         .WORD  36-'0
47 005312      052      052 .BYTE  '*', '*'    ; *
48 005314 177763         .WORD  -15
49 005316      045      045 .BYTE  '%, '%'    ; %
50 005320 177767         .WORD  -11
51 005322 000000         .WORD  0          ; END OF TABLE MARKER
52         .EVEN

```

SWVAL --- Accrue CSI switch value

```

1          .SDBTTL  SWVAL --- Accrue CSI switch value
2          ;-----
3          ; SWVAL is called to accrue a value associated with a CSI switch.
4          ; The value may be specified as Rad50 alphanumeric, decimal or octal.
5          ;
6          ; Inputs:
7          ; R1 = Pointer to start of value.
8          ;
9          ; Outputs:
10         ; R0 = Value accrued.
11         ; R1 = Pointer to value delimiter.
12         ; C-flag set on return if an error detected.
13         ;
14 005324 010546 SWVAL:  MOV     R5, -(SP)
15 005326 010105         MOV     R1, R5           ; SAVE POINTER TO START OF VALUE
16         ;
17         ; Determine what type of value this is.
18         ;
19 005330 004737 004660'        CALL    CSIGET       ; GET FIRST CHAR OF VALUE
20 005334 120027 000101        CMPB   RO, #'A       ; ALPHABETIC?
21 005340 103414                BLD    1$           ; BR IF NOT
22 005342 120027 000132        CMPB   RO, #'Z       ;
23 005346 101050                BHI    5$           ; BR IF NOT ALPHABETIC
24         ;
25         ; Rad50 value.
26         ;
27 005350 010501                MOV     R5, R1       ; GET BACK POINTER TO START OF VALUE
28 005352 010446                MOV     R4, -(SP)
29 005354 004737 005136'        CALL    GTRD50      ; ACCRUE A RAD50 VALUE
30 005360 010400                MOV     R4, R0       ; GET 1ST 3 CHARS OF VALUE
31 005362 012604                MOV     (SP)+, R4
32 005364 005705                TST    R5           ; 2ND 3 CHARS SHOULD BE NULL
33 005366 001436                BEQ    10$          ; BR IF OK
34 005370 000437                BR     5$           ; VALUE TOO LONG
35         ;
36         ; Value is not Rad50.
37         ; Make sure it is numeric.
38         ;
39 005372 120027 000055        1$:    CMPB   RO, #'-   ; ALLOW LEADING MINUS SIGN
40 005376 001406                BEQ    2$           ;
41 005400 120027 000060        CMPB   RO, #'0       ; IS VALUE NUMERIC?
42 005404 103431                BLD    5$           ; BR IF NOT
43 005406 120027 000071        CMPB   RO, #'9       ;
44 005412 101026                BHI    5$           ; BR IF NOT NUMERIC
45         ;
46         ; Value is numeric.
47         ; See if a decimal point was specified with value.
48         ;
49 005414 004737 004660'        2$:    CALL    CSIGET       ; GET NEXT CHAR IN VALUE
50 005420 120027 000056        CMPB   RO, #'.'       ; DECIMAL POINT?
51 005424 001413                BEQ    4$           ; BR IF YES
52 005426 120027 000060        CMPB   RO, #'0       ; ARE WE STILL PASSING DIGITS?
53 005432 103403                BLD    3$           ; BR IF NOT
54 005434 120027 000071        CMPB   RO, #'9       ;
55 005440 101765                BLOS  2$           ; KEEP LOOKING FOR DEC PT
56         ;
57         ; Octal value.

```

SWVAL -- Accrue CSI switch value

```
58 ;
59 005442 010501 3#: MOV R5,R1 ;GET POINTER TO START OF VALUE
60 005444 004737 005572' CALL ACROCT ;ACCRUE AN OCTAL VALUE
61 005450 103407 BCS 5# ;BR IF INVALID
62 005452 000404 BR 10#
63 ;
64 ; Decimal value.
65 ;
66 005454 010501 4#: MOV R5,R1 ;GET POINTER TO START OF VALUE
67 005456 004737 005476' CALL ACRDEC ;ACCRUE A DECIMAL VALUE
68 005462 005201 INC R1 ;POINT PAST DECIMAL POINT
69 ;
70 ; Successful finish
71 ;
72 005464 000241 10#: CLC ;SIGNAL GOOD RETURN
73 005466 000401 BR 11#
74 ;
75 ; Error
76 ;
77 005470 000261 5#: SEC ;SIGNAL ERROR ON RETURN
78 ;
79 ; Return
80 ;
81 005472 012605 11#: MOV (SP)+,R5
82 005474 000207 RETURN
```

ACRDEC -- Accrue a decimal value

```

1          .SBTTL  ACRDEC -- Accrue a decimal value
2          ;-----
3          ; ACRDEC is called to accrue a decimal character string.
4          ;
5          ; Inputs:
6          ;   R1 = Pointer to start of string.
7          ;
8          ; Outputs:
9          ;   R0 = Value accrued.
10         ;   R1 = Pointer to delimiter hit.
11         ;
12 005476 010346 ACRDEC: MOV     R3,-(SP)
13 005500 010446      MOV     R4,-(SP)
14 005502 005003      CLR     R3          ;FORM VALUE IN R3
15 005504 005004      CLR     R4          ;SET NON-NEGATIVE FLAG
16 005506 004737 004660' CALL   CSIGET      ;GET 1ST CHARACTER OF VALUE STRING
17 005512 120027 000055  CMPB   R0,#'-'    ;LEADING MINUS SIGN?
18 005516 001003      BNE     3$          ;BR IF NOT
19 005520 005204      INC     R4          ;SET NEGATIVE-FLAG
20 005522 004737 004660' 1$: CALL   CSIGET      ;GET NEXT CHARACTER
21 005526 120027 000071 3$:  CMPB   R0,#'9    ;IS THIS A DECIMAL DIGIT?
22 005532 101007      BHI     2$          ;BR IF NOT
23 005534 162700 000060  SUB     #'0,R0
24 005540 002404      BLT     2$          ;BR IF NOT DIGIT
25 005542 070327 000012  MUL     #10.,R3   ;MUL PREVIOUS VALUE BY 10.
26 005546 060003      ADD     R0,R3    ;ADD IN NEW DIGIT
27 005550 000764      BR      1$          ;GO GET NEXT DIGIT
28 005552 005301      2$: DEC     R1          ;POINT TO DELIMITER
29 005554 010300      MOV     R3,R0    ;RETURN RESULT IN R0
30 005556 005704      TST     R4          ;IS VALUE TO BE NEGATED?
31 005560 001401      BEQ     4$          ;BR IF NOT
32 005562 005400      NEG     R0          ;MAKE IT NEGATIVE
33 005564 012604      4$: MOV     (SP)+,R4
34 005566 012603      MOV     (SP)+,R3
35 005570 000207      RETURN

```

```

1          .SBTTL ACROCT -- Accrue an octal value
2          ;-----
3          ; ACROCT is called to accrue an octal character string.
4          ;
5          ; Inputs:
6          ; R1 = Pointer to start of string.
7          ;
8          ; Outputs:
9          ; R0 = Value accrued.
10         ; R1 = Pointer to delimiter hit.
11         ; C-flag set if error detected.
12         ;
13 005572 010346 ACROCT: MOV      R3, -(SP)
14 005574 010446          MOV      R4, -(SP)
15 005576 005003          CLR      R3          ; FORM VALUE IN R3
16 005600 005004          CLR      R4          ; SET NON-NEGATIVE FLAG
17 005602 004737 004660' CALL    CSIGET       ; GET 1ST CHAR OF VALUE STRING
18 005606 120027 000055  CMPB   R0, #'-'      ; LEADING MINUS FLAG?
19 005612 001003          BNE     4$          ; BR IF NOT
20 005614 005204          INC     R4          ; SET NEGATIVE FLAG
21 005616 004737 004660' 1$:  CALL    CSIGET       ; GET NEXT CHARACTER
22 005622 120027 000071  4$:  CMPB   R0, #'9      ; DIGIT?
23 005626 101013          BHI     2$          ; BR IF NOT
24 005630 120027 000067  CMPB   R0, #'7      ; OCTAL DIGIT
25 005634 101017          BHI     3$          ; BR IF INVALID OCTAL DIGIT
26 005636 162700 000060  SUB    #'0, R0      ; DIGIT?
27 005642 002405          BLT     2$          ; BR IF NOT
28 005644 006303          ASL     R3          ; MULTIPLY PREVIOUS VALUE BY 8.
29 005646 006303          ASL     R3
30 005650 006303          ASL     R3
31 005652 060003          ADD    R0, R3      ; ADD IN NEW DIGIT
32 005654 000760          BR     1$          ; GO GET NEXT DIGIT
33         ; Got valid value.
34 005656 010300 2$:  MOV      R3, R0      ; RETURN VALUE IN R0
35 005660 005704          TST    R4          ; IS VALUE TO BE NEGATED?
36 005662 001401          BEQ    6$          ; BR IF NOT
37 005664 005400          NEG    R0          ; MAKE VALUE NEGATIVE
38 005666 005301 6$:  DEC     R1          ; POINT TO DELIMITER
39 005670 000241          CLC                    ; SIGNAL NO ERROR
40 005672 000401          BR     5$
41         ; Invalid number
42 005674 000261 3$:  SEC                    ; SIGNAL ERROR ON RETURN
43 005676 012604 5$:  MOV      (SP)+, R4
44 005700 012603          MOV      (SP)+, R3
45 005702 000207          RETURN

```

UPUSH -- Push value onto user's stack

```

1          .SBTTL  UPUSH  -- Push value onto user's stack
2          ;-----
3          ; UPUSH is called during CSI processing to push a value onto the
4          ; user's stack.
5          ;
6          ; Inputs:
7          ;   R0 = Value to be pushed.
8          ;
9          ; Outputs:
10         ;   Value pushed onto user's stack.
11         ;
12 005704 010146  UPUSH:  MOV     R1, -(SP)
13 005706 106506      MFPD   SP             ; GET USER'S SP
14 005710 012601      MOV     (SP)+, R1
15 005712 010046      MOV     R0, -(SP)       ; PUSH VALUE ONTO USER'S STACK
16 005714 106641      MTPD   -(R1)
17 005716 010146      MOV     R1, -(SP)       ; STORE UPDATED STACK POINTER
18 005720 106606      MTPD   SP
19 005722 012601      MOV     (SP)+, R1
20 005724 000207      RETURN
21
22          000001      .END

```

Errors detected: 0

\*\*\* Assembler statistics

Work file reads: 0  
Work file writes: 0  
Size of work file: 9919 Words ( 39 Pages)  
Size of core pool: 17920 Words ( 70 Pages)  
Operating system: RT-11

Elapsed time: 00:00:33.20  
DK: TSEM3, LP: TSEM3=DK: TSEM3. MAC/C/N: SYM

#DILUP	1-40	5-25											
#INCDR	1-50	21-52	22-41										
#INKMN	1-46	14-27											
#NOVLN	1-44	3-115	3-116	3-119	3-120								
#VNOTT	1-35	2-18											
... V1	10-35	23-35	23-35	23-63	23-63	24-12	24-12	26-61	26-95	26-95	26-95	26-102	
	26-102	26-102	28-44										
... V2	10-35	10-35#	23-35	23-35	23-35#	23-35#	23-63	23-63	23-63#	23-63#	24-12	24-12#	
	26-95	26-95	26-95#	26-95#	26-102	26-102	26-102#	26-102#	28-44	28-44#			
ACRDEC	30-72	33-67	34-12#										
ACRFIL	26-55	30-17#											
ACROCT	33-60	35-13#											
ASKLIN	1-43	24-55											
ASNTBL	1-38	7-34											
AT##SZ	1-40	7-35											
BADEMT	1-46	11-23											
BADSWT	28-20#												
BADVAL	27-20	28-24#											
C. CSW	1-36	10-26											
CFPNT	1-42	24-47	24-48*	24-58*									
CHKABT	1-50	18-28											
CHNSIZ	1-52	10-21	19-29										
CORUSR	1-50	3-111	18-19	18-33	18-44								
CP#STD	1-47	15-26											
CPYCXD	5-43	6-16#											
CPYCXT	4-27	7-36	7-42	7-45	7-51	7-54	7-56	7-58	7-60	8-22	8-27	8-42	
	9-11#												
CPYEMT	1-31	5-16#											
CPYFIL	5-47	7-13#											
CPYMNT	1-39	7-68											
CPYPRV	5-55	8-9#											
CQ#CP	1-47	15-26*											
CQ#JOB	1-47	15-22*											
CQ#PRI	1-48	15-24*											
CQ#RO	1-47	15-19*											
CQ#R1	1-47	15-20*											
CQ#RNS	1-48	15-25*	15-29*										
CQ#RTN	1-47	15-23*											
CS#NMX	1-36	10-29											
CS#OPN	1-36	10-27											
CSIARE	1-44	7-22	26-61	26-85	26-95	26-102							
CSIBUF	1-42	24-33	24-35	24-54	24-69	25-30							
CSICH1	26-5#	27-31											
CSICHR	25-40	26-4#	26-22	26-105	27-39	27-48							
CSICOM	24-6	24-22#											
CSIDEV	1-43	25-29*	26-21*	30-23	30-46*								
CSIEQL	1-43	25-27*	25-44*	26-13	26-16*	26-37							
CSIER1	28-22	28-26	28-32#										
CSIERR	28-6	28-9	28-12	28-15	28-18	28-30#							
CSIFIL	1-44	26-61	26-72	26-83	26-95	26-95	26-102	30-22	30-25	30-27*	30-44*	30-53*	
	30-54#	30-62*	30-73*										
CSIFIN	26-7	27-53#											
CSIFUL	26-100	28-14#											
CSIGCL	24-28#	28-47											
CSIGEN	1-30	24-7#											
CSIGET	26-4	27-4	27-12	27-28	29-15#	29-18	29-20	30-31	31-17	31-32	33-19	33-49	



LITIME	1-48	15-27						
LMONHD	1-45	12-40	12-41*	14-12	17-21			
LPARNT	1-40	5-32	5-53					
LPRI	1-48	2-17*	15-24					
LSTSL	1-46	5-23	12-11	13-12	16-12			
LSW	1-50	2-18	5-25	21-52	22-41			
LSW2	1-45	3-115*	3-119*					
LSW2S	1-45	3-116*	3-120*					
LSW4	1-46	14-27						
LSW9	1-45							
LSWPBK	1-51	23-21						
MAXACC	1-38	7-44						
MAXASN	1-38	7-35						
MAXCXX	6-28	6-57#						
MAXLD	1-39	7-50	7-53					
MONO	11-28	12-5#						
MON1	11-29	13-5#						
MON2	11-30	14-6	14-8#					
MON3	11-31	14-5#						
MONABT	1-30	13-22	16-8#					
MONEMT	1-30	11-15#						
MONFGH	1-45	12-22	12-26*	17-28	17-29*			
MONFUN	11-24	11-28#	11-32					
MONQUE	14-38	14-46	15-10#					
MONREL	12-18	13-17	16-16	17-9#				
MVSIZ	1-52	20-17						
MXJPRI	1-41	2-13	2-15	8-40				
MXMONF	11-21	11-32#						
NLCHN	1-36	10-10						
NUCHN	1-38							
OF##SZ	1-39	7-44						
OKFILE	1-38	7-43						
OVRHC	1-43	2-20	7-68	24-55	28-38			
P0#SPV	1-41	3-23	3-78					
P0#SYS	1-40	14-5						
P2#CXT	1-40	5-34						
P2#VIR	1-40	3-117						
PR7	1-51							
PRGLDC	7-18	10-5#						
PRIVA0	1-41	3-45	3-65*	3-81	3-92*	8-20		
PRIVC0	1-41	3-23	3-39	3-58*	3-78	3-85*	8-33*	14-5
PRIVC2	1-40	3-117	5-34					
PRIVF0	1-35	3-62*	3-89*	8-34*				
PRIVS0	1-41	3-42	3-61*	3-88*	8-26	8-33	8-34	
PS	1-22#	18-16*	18-25*	18-34*				
PUTUCH	1-43	24-72	24-74					
PVNPW	1-41	3-46	3-67	3-94	8-21	8-31		
QCOMPL	1-48	15-34						
QNSPNX	1-50	18-27						
R#CHN	1-52	10-16	10-20	19-24	19-28			
R#XCHN	1-52	10-20	19-28					
R5OCHR	30-35	31-18	31-33	32-14#				
R5ODK	1-94#	25-29	26-21					
R5ORNG	32-15	32-43#						
RC##SZ	1-36	21-47						
RC#BAS	1-37	21-65						



... CM0	26-61											
... CM1	23-35	23-63	26-95	26-102								
... CM2	23-35	23-35	23-35	23-35	23-63	23-63	23-63	23-63	26-95	26-95	26-95	26-102
	26-102											
... CM3	10-35	24-12	28-44									
... CM5	23-35	23-63	26-61	26-95	26-102							
... CM7	23-35	23-63										
. CLOSE	1-26#	24-12										
. DSTAT	1-26#	26-61										
. ENTER	1-26#	26-95										
. LOOKU	1-26#	26-102										
. PURGE	1-26#	10-35	28-44									
. READW	1-26#	23-35	23-63									
DIE	1-72#											
DISABL	1-57#	18-16										
ENABL	1-61#	18-25	18-34									
OCALL	1-83#	2-20	7-68	24-55	28-38							