Table of contents

```
                                        .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.
                                ;
000000                                  .CSECT  TSEM3
000000  020551                  TSEM2:  .RAD50  /EM3/            ;Overlay id
        177776                  PS      =       177776          ;Processor Status Word
                                ;
                                ;  Macro calls
                                ;
                                        .MCALL  .CLOSE,.DSTATUS,.ENTER,.LOOKUP,.PURGE,.READW
                                ;
                                ;  Global definitions
                                ;
                                        .GLOBL  CSISPC,CSIGEN,MONEMT,MONABT,GETCXT,FRECXT,GTUKBL
                                        .GLOBL  GETCHA,SETPRI,CPYEMT,KMCPCX,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,UREGO,$INCOR,LSW,CXBMOV
                                        .GLOBL  CXTBAS,LSWPBK,USREMT,PR7,INTPRI,CXTRMN
                                        .GLOBL  R$CHN,R$XCHN,CHNSIZ,MVSIZ

                                ;-------------------------------------------------------
                                ;  Macros to enable and disable interrupts.
                                ;
                                        .MACRO  DISABL          ;DISABLE INTERRUPTS
```

```
                        BIS     #340,@#PS
                        .ENDM   DISABL
                ;
                        .MACRO  ENABL           ;ENABLE INTERRUPTS
                        BIC     INTPRI,@#PS
                        .ENDM   ENABL
                ;
                ;  Macro to print an error message when a system crash occurs.
                ;
                ;  Arguments:
                ;   MSG = Name of error message to print.
                ;   ARG = (Optional) argument value to display with error message.
                ;
                        .GLOBL  DIEMSG,DIEARG,SYSHLT
                        .MACRO  DIE     MSG,ARG
                        MOV     MSG,@#DIEMSG
                        .IF     NB,ARG
                        MOV     ARG,@#DIEARG
                        .ENDC
                        CALL    @#SYSHLT
                        .ENDM   DIE
                ;
                ;  Macro definition for calling global routines residing in mapped
                ;  system regions.
                ;
                        .MACRO  OCALL   ENTADD
                        .IF     B,ENTADD
                                .ERROR  ;OCALL SPECIFIED WITH NO ENTRY ADDRESS
                                .MEXIT
                        .ENDC
                        CALL    OVRHC           ;CALL THE OVERLAY HANDLER
                        .WORD   ENTADD          ;SPECIFY THE ENTRY POINT
                        .ENDM
                ;-------------------------------------------------------------------------------
                ;  Data areas
                ;
000002  015270  R50DK:  .RAD50  /DK /
```

```
                                        .SBTTL  SETPRI -- Set priority or privilege for job
                                ;-------------------------------------------------------------------
                                ;  Set execution priority or privilege flags for job.
                                ;
000004  105737  000000G        SETPRI: TSTB    EMTBLK          ;Set priority or privilege?
000010  001027                        BNE     SETPRV          ;Set privilege
                                ;
                                ;  Set job execution priority
                                ;
000012  013702  0000020                MOV     EMTBLK+2,R2     ;Get specified execution priority
000016  003002                        BGT     1$              ;Br if value specified
000020  113702  000000G               MOVB    VPRIDF,R2       ;If not specified, set default
000024  120237  000000G        1$:     CMPB    R2,MXJPRI       ;Compare with max authorized priority
000030  101402                        BLOS    2$              ;Br if ok
000032  113702  000000G               MOVB    MXJPRI,R2       ;Get max authorized priority
000036  110261  000000G        2$:     MOVB    R2,LBSPRI(R1)   ;Set base priority for job
000042  110261  000000G               MOVB    R2,LPRI(R1)     ;Set current priority for job
000046  032761  000000G 000000G       BIT     #$VNOTT,LSW(R1) ;Is this a subprocess not connected to term?
000054  001403                        BEQ     9$              ;Br if not
000056                                OCALL   SPPRED          ;Reduce priority of disconnected subprocesses
000064  000137  000000G        9$:     JMP     EMTXIT          ;Finished
```

```
                                ;-------------------------------------------------------------------------------
                                ;  Set privilege flags for job.
                                ;
                                ;  The form of the EMT argument block is as follows:
                                ;
                                ;       .BYTE   1,150
                                ;       .BYTE   fun,perm        ;Fun: 0=get,1=clear,2=set  Perm: 0=temp,1=perm
                                ;       .WORD   buffer          ;Buffer address
                                ;       .WORD   0               ;(reserved)
                                ;
000070                          SETPRV:
                                ;
                                ;  Validate the user's buffer address
                                ;
000070  013702  000040                  MOV     EMTBLK+4,R2     ;Get user's buffer address
000074  010200                          MOV     R2,R0
000076  004737  000000                  CALL    VALADW          ;Validate the address
                                ;
                                ;  Only allow changes to authorization privilege flags if SETPRV is enabled.
                                ;
000102  123727  000030 000002          CMPB    EMTBLK+3,#2     ;Does he want to access auth privileges?
000110  001012                         BNE     10$             ;Br if not
000112  032737  000000 000000          BIT     #P0$SPV,PRIVC0  ;Does he have SETPRV privilege?
000120  001006                         BNE     10$             ;Br if yes
000122  123727  000020 000002          CMPB    EMTBLK+2,#2     ;Does he want to set bits?
000130  001002                         BNE     10$             ;Br if not (allow access or clear-bits)
000132  105337  000030                  DECB    EMTBLK+3        ;Can't change auth priv without SETPRV
                                ;
                                ;  See what the subfunction is
                                ;
000136  113700  000020          10$:    MOVB    EMTBLK+2,R0     ;Get subfunction code
000142  020027  000001                  CMP     R0,#1           ;Clear bits?
000146  001424                         BEQ     2$              ;Br if yes
000150  020027  000002                  CMP     R0,#2           ;Set bits?
000154  001453                         BEQ     5$              ;Br if yes
                                ;
                                ;  Return current privilege flags to user
                                ;
000156  012703  000000                  MOV     #PRIVC0,R3      ;Point to vector with current flags
000162  113700  000030                  MOVB    EMTBLK+3,R0     ;Get type of priv flags wanted
000166  001406                         BEQ     11$             ;Br if wants current priv flags
000170  012703  000000                  MOV     #PRIVS0,R3      ;Point to table of perm flags
000174  005300                         DEC     R0              ;Want perm flags?
000176  001402                         BEQ     11$             ;Br if yes
000200  012703  000000                  MOV     #PRIVA0,R3      ;Must want authorized privileges
000204  012700  000000          11$:    MOV     #PVNPW,R0       ;Get # words to move
000210  012346                  1$:     MOV     (R3)+,-(SP)     ;Get word with current flags
000212  106622                         MTPD    (R2)+           ;Move to user's buffer
000214  077003                         SOB     R0,1$           ;Move all words with privilege flags
000216  000504                         BR      9$
                                ;
                                ;  User wants to clear some privilege flags
                                ;
000220  005005                  2$:     CLR     R5              ;Init index into privilege words
000222  113704  000030                  MOVB    EMTBLK+3,R4     ;Get type of privilege change wanted
000226  106522                  3$:     MFPD    (R2)+           ;Get flags from user
000230  012600                          MOV     (SP)+,R0        ;Pop flags
```

```
     58 000232  040065  000000G                    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  000000G                    BIC     R0,PRIVS0(R5)   ;Change permanent flags too
     62 000246  040065  000000G                    BIC     R0,PRIVF0(R5)   ;Change command file privileges
     63 000252  020427  000002                     CMP     R4,#2           ;Change auth flags?
     64 000256  001002                             BNE     4$              ;Br if not
     65 000260  040065  000000G                    BIC     R0,PRIVA0(R5)   ;Change authorization flags too
     66 000264  062705  000002          4$:        ADD     #2,R5           ;Get index to next priv word
     67 000270  020527  000000C                    CMP     R5,#PVNPW*2     ;Done all words?
     68 000274  103754                             BLO     3$              ;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  000000G 000000G            BIT     #PO$SPV,PRIVC0  ;Is user allowed to set any flag?
     79 000322  001006                             BNE     6$              ;Br if yes
     80 000324  010400                             MOV     R4,R0           ;Get flags to set
     81 000326  046500  000000G                    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  000000G         6$:        BIS     R4,PRIVC0(R5)   ;Set current flags
     86 000344  105737  000003G                    TSTB    EMTBLK+3        ;Is change to be permanent?
     87 000350  001412                             BEQ     7$              ;Br if not
     88 000352  050465  000000G                    BIS     R4,PRIVS0(R5)   ;Make permanent change
     89 000356  050465  000000G                    BIS     R4,PRIVF0(R5)   ;Change command file privileges
     90 000362  123727  000003G 000002            CMPB    EMTBLK+3,#2     ;Change auth flags too?
     91 000370  001002                             BNE     7$              ;Br if not
     92 000372  050465  000000G                    BIS     R4,PRIVA0(R5)   ;Change authorization privileges
     93 000376  062705  000002          7$:        ADD     #2,R5           ;Get index to next priv word
     94 000402  020527  000000C                    CMP     R5,#PVNPW*2     ;More priv words to change?
     95 000406  103740                             BLO     8$              ;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  000001                     MOV     #1,R0           ;Return error code 1
    100 000424  000137  000000G                    JMP     SETERR
    101                               ;
    102                               ;  Finished
    103                               ;
    104 000430  000137  000000G         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  000000G                   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
```

```
                                        .SBTTL  KMCPCX -- Kmon EMT to copy context block data
                                ;-----------------------------------------------------------------------
                                ;  Tskmon EMT to copy data from the context block of another job into the
                                ;  context block of our job.
                                ;
                                ;  Form of EMT argument block:
                                ;
                                ;       .BYTE   0,126
                                ;       .WORD   14
                                ;       .WORD   job_number
                                ;       .WORD   address_of_item
                                ;       .WORD   num_bytes_to_copy
                                ;
000506                          KMCPCX:
                                ;
                                ;  Gain exclusive access to context block buffer
                                ;
000506  004737  002324'                CALL    GETCXT          ;Gain access to context buffer
000512  103002                         BCC     1$              ;Br if got it
000514  004737  000000G                CALL    STOP            ;Job was aborted
                                ;
                                ;  Copy the requested data
                                ;
000520  013702  000004G         1$:     MOV     EMTBLK+4,R2     ;Get # of job we are copying from
000524  013703  000006G                 MOV     EMTBLK+6,R3     ;Get address of item we are copying
000530  013705  000010G                 MOV     EMTBLK+8.,R5    ;Get # bytes to copy
000534  004737  001342'                CALL    CPYCXT          ;Copy the data
                                ;
                                ;  Finished
                                ;
000540  004737  002410'                CALL    FRECXT          ;Free the context buffer
000544  000137  000000G                JMP     EMTXIT
```

```
                                        .SBTTL  CPYEMT -- Copy file/privilege info from another job
                                ;-----------------------------------------------------------------------
                                ;  This EMT is used to copy the file and privilege information from another
                                ;  job.  The form of the EMT argument block is:
                                ;
                                ;       .BYTE   subfun,160
                                ;       .WORD   jobnum
                                ;       .WORD   0
                                ;
                                ;  Where subfun has the following meanings:
                                ;   0 ==> Copy file assess information only.
                                ;   1 ==> Copy file info and privilege info.
                                ;   3 ==> Copy info from context block into program buffer.
                                ;  Jobnum is the number of the job that we are copying from.
                                ;
000550                          CPYEMT:
                                ;
                                ;  See if the specified job is logged on
                                ;
000550  013702  0000020               MOV     EMTBLK+2,R2     ;Get # of job we are copying from
000554  006302                        ASL     R2              ;Convert to job index number
000556  001407                        BEQ     10$             ;Br if job number 0
000560  020227  0000000               CMP     R2,#LSTSL       ;Valid job number?
000564  101004                        BHI     10$             ;Br if not
000566  032762  0000000 0000000       BIT     ##$DILUP,LSW(R2) ;Is job logged on?
000574  001004                        BNE     1$              ;Br if yes
000576  012700  000001          10$:  MOV     #1,R0           ;Return error code 1
000602  000137  0000000               JMP     SETERR
                                ;
                                ;  See if job is privileged to do this
                                ;
000606  026102  0000000         1$:   CMP     LPARNT(R1),R2   ;Are we copying info from our parent job?
000612  001407                        BEQ     4$              ;Br if yes -- That is always legal
000614  032737  0000000 0000000       BIT     #P2$CXT,PRIVC2  ;Are we authorized to copy context info?
000622  001003                        BNE     4$              ;Br if yes
000624  005000                3$:     CLR     R0              ;Return error code 0
000626  000137  0000000               JMP     SETERR
                                ;
                                ;  See if we are copying into our context block or into program buffer
                                ;
000632  123727  0000000 000003  4$:   CMPB    EMTBLK,#3       ;Copy data into program buffer?
000640  001002                        BNE     2$              ;Br if not
000642  000137  000676'               JMP     CPYCXD          ;Copy context data into program buffer
                                ;
                                ;  Copy the file access info
                                ;
000646  004737  001022'         2$:   CALL    CPYFIL          ;Copy file access info for job
                                ;
                                ;  Copy the privilege info
                                ;
000652  105737  0000000               TSTB    EMTBLK          ;Does he want to copy privileges?
000656  001405                        BEQ     9$              ;Br if not
000660  026102  0000000               CMP     LPARNT(R1),R2   ;Copying privileges from parent?
000664  001357                        BNE     3$              ;Br if not
000666  004737  001222'               CALL    CPYPRV          ;Copy privilege info
                                ;
                                ;  Finished
```

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

```
                                        .SBTTL  CPYCXD -- EMT to get information from another jobs context
                                ;-------------------------------------------------------------------------
                                ;  Copy data from the context block of a job into a program buffer area.
                                ;
                                ;  Inputs:
                                ;   R1 = Index number of our job
                                ;   R2 = Index number of job we are copying from
                                ;
                                ;  EMT argument block:
                                ;
                                ;       .BYTE   3,160
                                ;       .WORD   job_number
                                ;       .WORD   item_index
                                ;       .WORD   buffer_address
                                ;
000676                          CPYCXD:
                                ;
                                ;  Gain exclusive access to context buffer
                                ;
000676  004737  002324'                 CALL    GETCXT          ;Gain access to context buffer
000702  103002                         BCC     1$              ;Br if got it
000704  004737  000000G                 CALL    STOP            ;Aborted while waiting
                                ;
                                ;  Get info about data to be copied
                                ;
000710  013704  000040G         1$:     MOV     EMTBLK+4,R4     ;Get index # of item of interest
000714  006304                          ASL     R4              ;Convert to word table index
000716  020427  000004                  CMP     R4,#MAXCXX      ;Is it too big?
000722  101404                          BLOS    2$              ;Br if ok
000724  012700  000002                  MOV     #2,R0           ;Return error 2 if too big
000730  000137  000000G                 JMP     SETERR
000734  016403  001006'         2$:     MOV     CXXADR(R4),R3   ;Get address of start of data
000740  016404  001014'                 MOV     CXXSIZ(R4),R4   ;Get # bytes wanted
000744  010400                          MOV     R4,R0
000746  004737  002726'                 CALL    REDCXT          ;Read data into context buffer
                                ;
                                ;  Move data to user's buffer
                                ;
000752  013700  000060G                 MOV     EMTBLK+6,R0     ;Get address of user's buffer
000756  004737  000000G                 CALL    VALADW          ;Make sure buffer is ok
000762  013705  000000G                 MOV     CXTBUF,R5       ;Get address of buffer
000766  006204                          ASR     R4              ;Get # words to move
000770  012546                  3$:     MOV     (R5)+,-(SP)     ;Get next word
000772  106620                          MTPD    (R0)+           ;Move to user's buffer
000774  077403                          SOB     R4,3$           ;Loop if more to move
                                ;
                                ;  Finished
                                ;
000776  004737  002410'                 CALL    FRECXT          ;Free context buffer
001002  000137  000000G                 JMP     EMTXIT
                                ;
                                ;  Table of addresses of items in context block
                                ;
001006  000000G                 CXXADR: .WORD   JPWDEV           ; 0 - Name of print-window device
001010  000000G                         .WORD   JPWTYP           ; 1 - Print-window device type
001012  000000G                         .WORD   JPWFLG           ; 2 - Print-window flag word
        000004                  MAXCXX  =       .-CXXADR-2
```

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

```
                                        .SBTTL  CPYFIL -- Copy file access info from another job
                                ;-------------------------------------------------------------------------
                                ;  CPYFIL is called to copy all of the file access information from
                                ;  another job to our job.  The information copied is as follows:
                                ;  1. ASSIGN commands.
                                ;  2. Logical disk information.
                                ;  3. Mounted devices information.
                                ;  4. ACCESS control information.
                                ;
                                ;  Inputs:
                                ;   R2 = Job index number of job whose file info is to be copied.
                                ;
001022  010346                  CPYFIL: MOV     R3,-(SP)
001024  010546                          MOV     R5,-(SP)
                                ;
                                ;  Purge all channels that are open to logical disks
                                ;
001026  004737  001372'                 CALL    PRGLDC          ;Purge all LD channels
                                ;
                                ;  Dismount all mounted devices for this job
                                ;
001032  012700  000000G                 MOV     #CSIARE,R0      ;Point to internal EMT arg block area
001036  012710  056402                  MOV     #2+<135*400>,(R0);Set EMT function code
001042  104375                          EMT     375             ;Dismount all devices for job
                                ;
                                ;  Gain exclusive access to job context buffer
                                ;
001044  004737  002324'                 CALL    GETCXT          ;Get access to context buffer
001050  103002                          BCC     3$              ;Br if got it
001052  004737  000000G                 CALL    STOP            ;Job was aborted while waiting
                                ;
                                ;  Set up ASSIGN information
                                ;
001056  012703  000000G         3$:     MOV     #ASNTBL,R3      ;Point to assign table
001062  012705  000000C                 MOV     #MAXASN*AT$$SZ,R5;# bytes to get
001066  004737  001342'                 CALL    CPYCXT          ;Copy data into our context block
                                ;
                                ;  Set up ACCESS information
                                ;
001072  012703  000000G                 MOV     #RESDEV,R3      ;Get this control word
001076  012705  000002                  MOV     #2,R5           ;2 bytes for this
001102  004737  001342'                 CALL    CPYCXT          ;Copy data into our context block
001106  012703  000000G                 MOV     #OKFILE,R3      ;Now get actual access table
001112  012705  000000C                 MOV     #MAXACC*OF$$SZ,R5 ;# bytes in table
001116  004737  001342'                 CALL    CPYCXT          ;Copy data into our context block
                                ;
                                ;  Set up logical disk information
                                ;
001122  012703  000000G                 MOV     #LDNAME,R3      ;Copy each LD control table
001126  012705  000000C                 MOV     #8.*MAXLD,R5
001132  004737  001342'                 CALL    CPYCXT
001136  012703  000000G                 MOV     #LDPDEV,R3
001142  012705  000000C                 MOV     #2*MAXLD,R5
001146  004737  001342'                 CALL    CPYCXT
001152  012703  000000G                 MOV     #LDSIZE,R3
001156  004737  001342'                 CALL    CPYCXT
001162  012703  000000G                 MOV     #LDBASE,R3
```

```
     58 001166  004737  001342'                    CALL    CPYCXT
     59 001172  012703  000000                     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
```

```
                                        .SBTTL  CPYPRV -- Copy privilege flags from another job's context
                                ;-----------------------------------------------------------------------
                                ;  Copy the authorized and set privilege flags from another job's context
                                ;  block to our context block.
                                ;
                                ;  Inputs:
                                ;   R2 = Index number of job we are copying from.
                                ;
001222  010346                  CPYPRV: MOV     R3,-(SP)
001224  010546                          MOV     R5,-(SP)
                                ;
                                ;  Gain exclusive access to context buffer
                                ;
001226  004737  002324'                 CALL    GETCXT          ;Gain access to context buffer
001232  103002                          BCC     3$              ;Br if got it
001234  004737  0000000                 CALL    STOP            ;Job was aborted
                                ;
                                ;  Copy authorized privileges
                                ;
001240  012703  0000000         3$:     MOV     #PRIVAO,R3      ;Point to authorized privilege vector
001244  012705  000000C                 MOV     #PVNPW*2,R5     ;Get # bytes to copy
001250  004737  001342'                 CALL    CPYCXT          ;Copy context info
                                ;
                                ;  Copy set privileges
                                ;
001254  012703  0000000                 MOV     #PRIVSO,R3      ;Point to set privilege vector
001260  004737  001342'                 CALL    CPYCXT          ;Copy context into
                                ;
                                ;  Set current privileges to set privileges
                                ;
001264  012700  0000000                 MOV     #PVNPW,R0       ;Get # privilege words
001270  005005                          CLR     R5              ;Init index
001272  016565  0000000 0000000 1$:     MOV     PRIVSO(R5),PRIVCO(R5);Current privileges
001300  016565  0000000 0000000         MOV     PRIVSO(R5),PRIVFO(R5);Command file privileges
001306  062705  000002                  ADD     #2,R5           ;Increment word index
001312  077011                          SOB     R0,1$           ;Loop if more to copy
                                ;
                                ;  Copy maximum authorized priority
                                ;
001314  012703  0000000                 MOV     #MXJPRI,R3      ;Get maximum authorized priority
001320  012705  000001                  MOV     #1,R5           ;1 byte
001324  004737  001342'                 CALL    CPYCXT
                                ;
                                ;  Finished
                                ;
001330  004737  002410'                 CALL    FRECXT          ;Free context buffer
001334  012605                          MOV     (SP)+,R5
001336  012603                          MOV     (SP)+,R3
001340  000207                          RETURN
```

```
                        .SBTTL  CPYCXT -- Copy info from another job's context block
                ;-----------------------------------------------------------------------
                ;  CPYCXT is called to copy information from another job's context block
                ;  into our context block.
                ;
                ;  Inputs:
                ;   R2 = Index number of job that we are copying from.
                ;   R3 = Address within context block of info to be copied.
                ;   R5 = Number of bytes to copy.
                ;
001342  010346  CPYCXT: MOV     R3,-(SP)
001344  010546         MOV     R5,-(SP)
                ;
                ;  Read the info into the context block buffer
                ;
001346  010500         MOV     R5,R0           ;Get # bytes to be read
001350  004737  002726'        CALL    REDCXT          ;Move info into context buffer
                ;
                ;  Now move info from buffer to our context block
                ;
001354  013700  000000G        MOV     CXTBUF,R0       ;Point to buffer
001360  112023  1$:     MOVB    (R0)+,(R3)+     ;Copy data into our context area
001362  077502         SOB     R5,1$
                ;
                ;  Finished
                ;
001364  012605         MOV     (SP)+,R5
001366  012603         MOV     (SP)+,R3
001370  000207         RETURN
```

```
                        .SBTTL  PRGLDC -- Purge channels opened to logical disks
                ;-----------------------------------------------------------------------
                ;  PRGLDC is called to purge all channels that are opened to logical disks.
                ;
001372  010346 PRGLDC:  MOV     R3,-(SP)
001374  010546         MOV     R5,-(SP)
                ;
                ;  Begin loop through all channels
                ;
001376  012703  177777  MOV     #NLCHN-1,R3     ;Get # of last channel
                ;
                ;  Compute address of channel block for this channel
                ;
001402  010305  1$:     MOV     R3,R5           ;Get channel number
001404  013700  000000G MOV     CXTRMN,R0       ;Get address of base of simulated RMON
001410  062700  000000G ADD     #R$CHN,R0       ;Point to block for channel 0
001414  020527  000021  CMP     R5,#21          ;Is this channel in extended channel area?
001420  103404          BLO     2$              ;Br if not
001422  162705  000021  SUB     #21,R5          ;Get channel # relative to extended channels
001426  062700  000000C ADD     #R$XCHN-R$CHN,R0;Point to extended channel area
001432  070527  000000G 2$:     MUL     #CHNSIZ,R5      ;Multiply by # bytes per channel
001436  060005          ADD     R0,R5           ;Get address of channel block
                ;
                ;  See if this channel is opened to a logical disk
                ;
001440  016500  000000G MOV     C.CSW(R5),R0    ;Get status word for channel
001444  032700  000000G BIT     #CS$OPN,R0      ;Is this channel open?
001450  001411          BEQ     3$              ;Br if not
001452  042700  000000C BIC     #^C<CS$NMX>,R0  ;Extract device index number
001456  120037  000000G CMPB    R0,LDDEVX       ;Is chan opened to a logical disk?
001462  001004          BNE     3$              ;Br if not
                ;
                ;  Purge this channel
                ;
001464                  .PURGE  R3              ;Purge the channel
                ;
                ;  Check the next channel
                ;
001474  005303  3$:     DEC     R3              ;Get next channel number
001476  002341          BGE     1$              ;Loop if more to check
                ;
                ;  Finished
                ;
001500  012605          MOV     (SP)+,R5
001502  012603          MOV     (SP)+,R3
001504  000207          RETURN
```

```
                                        .SBTTL  Job monitoring EMT's
                                ;-------------------------------------------------------------------------
                                ;  The following EMT's are part of the TSX-Plus facility that allows one
                                ;  job to monitor the state of another job.
                                ;
                                ;  The form of the EMT is:
                                ;
                                ;       .BYTE   code,157
                                ;       .WORD   arg1
                                ;       .WORD   arg2
                                ;
                                ;  Where "code" is the sub-function code indicating which job-monitoring
                                ;  function is wanted.
                                ;
001506                          MONEMT:
                                ;
                                ;  Get sub-function code and see if it is legal.
                                ;
001506  113702  0000000                 MOVB    EMTBLK,R2       ;Get sub-function code
001512  006302                          ASL     R2              ;Convert to word table index
001514  020227  000010                  CMP     R2,#MXMONF      ;Is it legal?
001520  101402                          BLOS    1$              ;Br if ok
001522  000137  0000000                 JMP     BADEMT          ;Invalid sub-function code
001526  000172  001532'       1$:       JMP     @MONFUN(R2)     ;Enter processing routine based on sub-functn
                                ;
                                ;  Jump vector for sub-function codes
                                ;
001532  001542'         MONFUN: .WORD   MON0            ; 0 -- Begin monitoring a job
001534  001666'                 .WORD   MON1            ; 1 -- Stop monitoring a job
001536  001750'                 .WORD   MON2            ; 2 -- Send status to monitoring jobs
001540  001732'                 .WORD   MON3            ; 3 -- Send status from system program
000010                  MXMONF  =       .-MONFUN        ;Max sub-function index
```

```
                                ;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                                ;  Sub-function # 0
                                ;  Begin monitoring a specified job.
                                ;
001542                          MONO:
                                ;
                                ;  Get the number of the job to monitor and see if it is valid
                                ;
001542  013702  0000020                MOV     EMTBLK+2,R2     ;Get # of job to be monitored
001546  006302                     ASL     R2              ;Convert to job index number
001550  020227  0000000             CMP     R2,#LSTSL       ;Is the job number too large?
001554  101040                     BHI     8$              ;Br if yes
                                ;
                                ;  The job number is valid.
                                ;  If our job has previously set up monitoring on the same job we are
                                ;  about to monitor, release the previous monitoring.
                                ;
001556  004737  002234'             CALL    MONREL          ;Release any previous monitoring
                                ;
                                ;  Get a free monitoring control block
                                ;
001562  013703  0000000             MOV     MONFQH,R3       ;Get a free monitor control block
001566  001004                     BNE     2$              ;Br if got one
001570  012700  000002              MOV     #2,R0           ;Return error 2 if no free control blocks
001574  000137  0000000             JMP     SETERR
001600  016337  0000000 0000000 2$: MOV     JM$LNK(R3),MONFQH ;Remove block from free list
                                ;
                                ;  Initialize the monitoring control block
                                ;
001606  110163  0000000             MOVB    R1,JM$JOB(R3)   ;Set # of job that is monitoring
001612  013763  0000040 0000000     MOV     EMTBLK+4,JM$RTN(R3) ;Set address of completion routine
                                ;
                                ;  Add monitoring control block to list for job being monitored
                                ;
001620  005702                     TST     R2              ;Request to monitor all jobs?
001622  001006                     BNE     3$              ;Br if not
001624  013763  0000000 0000000     MOV     SMONHD,JM$LNK(R3);Add to list of requests for all jobs
001632  010337  0000000             MOV     R3,SMONHD
001636  000405                     BR      9$
001640  016263  0000000 0000000 3$: MOV     LMONHD(R2),JM$LNK(R3) ;Add to list for job being monitored
001646  010362  0000000             MOV     R3,LMONHD(R2)
                                ;
                                ;  Finished
                                ;
001652  000137  0000000         9$: JMP     EMTXIT
                                ;
                                ;  Error, invalid job number
                                ;
001656  012700  000001          8$: MOV     #1,R0           ;Return error code 1
001662  000137  0000000             JMP     SETERR
```

```
                                ;-------------------------------------------------------------------------------
                                ;  Subfunction 1.
                                ;  Stop monitoring a job.
                                ;
001666                          MON1:
                                ;
                                ;  Get the number of the job to monitor and see if it is valid
                                ;
001666  013702  0000020                 MOV     EMTBLK+2,R2     ;Get # of job to be monitored
001672  001407                          BEQ     1$              ;Zero ==> Release all monitors
001674  006302                          ASL     R2              ;Convert to job index number
001676  020227  0000000                 CMP     R2,#LSTSL       ;Is the job number too large?
001702  101007                          BHI     8$              ;Br if yes
                                ;
                                ;  Release monitoring a specific job
                                ;
001704  004737  002234'                 CALL    MONREL          ;Release monitoring the specified job
001710  000402                          BR      9$
                                ;
                                ;  Release monitoring all jobs
                                ;
001712  004737  002210'          1$:    CALL    MONABT          ;Release all job monitoring
                                ;
                                ;  Finished
                                ;
001716  000137  0000000          9$:    JMP     EMTXIT
                                ;
                                ;  Error -- Invalid job number specified
                                ;
001722  012700  000001           8$:    MOV     #1,R0           ;Return error code 1
001726  000137  0000000                 JMP     SETERR
```

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

```
                                ;-------------------------------------------------------------------------------
                                ;  MONQUE is called to queue a completion routine to pass a monitoring
                                ;  status message to a monitoring job.
                                ;
                                ;  Inputs:
                                ;   R3 = Pointer to monitor control block (JM$xxx)
                                ;   R5 = Job number and system status flag to be passed in R0
                                ;   EMTBLK+2 = Status code to be passed to completion routine in R1
                                ;
002076  010146                  MONQUE: MOV     R1,-(SP)
002100  010446                          MOV     R4,-(SP)
                                ;
                                ;  Get a free completion queue element.
                                ;
002102  004737  0000000G                CALL    GETQ            ;Get free queue element (addr in R1)
                                ;
                                ;  Set up information in the completion queue element
                                ;
002106  010561  0000000G                MOV     R5,CQ$R0(R1)    ;Pass job number in R0
002112  013761  0000002G 0000000G       MOV     EMTBLK+2,CQ$R1(R1) ;Pass status code in R1
002120  116300  0000000G                MOVB    JM$JOB(R3),R0   ;Get # of job who is monitoring us
002124  110061  0000000G                MOVB    R0,CQ$JOB(R1)   ;Completion routine is for that job
002130  016361  0000000G 0000000G       MOV     JM$RTN(R3),CQ$RTN(R1);Set address of completion routine
002136  116061  0000000G 0000000G       MOVB    LPRI(R0),CQ$PRI(R1);Set execution priority
002144  112761  0000000G 0000000G       MOVB    #S$IOFN,CQ$RNS(R1);Set job execution state
002152  112761  0000000G 0000000G       MOVB    #CP$STD,CQ$CP(R1);Set compl routine class priority
002160  005760  0000000G                TST     LITIME(R0)      ;Is this an interactive job?
002164  001403                          BEQ     2$              ;Br if not
002166  112761  0000000G 0000000G       MOVB    #S$HICP,CQ$RNS(R1);Set interactive state
002174  010104                  2$:     MOV     R1,R4           ;Get address of Q element to R4 for QCOMPL
                                ;
                                ;  Queue a completion routine for the monitoring job
                                ;
002176  004737  0000000G                CALL    QCOMPL          ;Queue a completion routine
                                ;
                                ;  Finished
                                ;
002202  012604                          MOV     (SP)+,R4
002204  012601                          MOV     (SP)+,R1
002206  000207                          RETURN
```

```
                                ;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                                ;  MONABT is called to release all job monitoring requests issued by a
                                ;  job.
                                ;
                                ;  Inputs:
                                ;   R1 = Job index number of job whose monitoring requests are to be released.
                                ;
002210  010246                  MONABT: MOV     R2,-(SP)
                                ;
                                ;  Begin loop to release requests for all jobs on the system.
                                ;
002212  012702  0000000                 MOV     #LSTSL,R2       ;Get # of last line
                                ;
                                ;  Release all monitoring being done on that job by our job
                                ;
002216  004737  002234'         1$:     CALL    MONREL          ;Release monitoring
                                ;
                                ;  Loop if more jobs to check
                                ;
002222  162702  000002                 SUB     #2,R2           ;Are there more jobs to check?
002226  002373                         BGE     1$              ;Loop if yes (0=monitor all jobs)
                                ;
                                ;  Finished
                                ;
002230  012602                         MOV     (SP)+,R2
002232  000207                         RETURN
```

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

```
                                        .SBTTL  GETCXT -- Obtain exclusive access to context buffer
                                ;-------------------------------------------------------------------
                                ;  This routine is called to obtain exclusive access to the buffer
                                ;  used to access the context block for another job (CXTBUF).
                                ;  On return, exclusive access has been granted to the context
                                ;  block buffer unless the job was aborted while waiting for access.
                                ;
                                ;   Outputs:
                                ;       C-flag cleared ==> Access granted.
                                ;       C-flag set ==> Job aborted while waiting for buffer.
                                ;
002324                          GETCXT:
                                ;
                                ;  See if the context buffer is currently free
                                ;
002324                          1$:     DISABL                  ;;;Disable interrupts
002332  113700  0000000G               MOVB    CXBOWN,R0       ;;;Is buffer currently free?
002336  001415                         BEQ     3$              ;;;Br if yes
002340  120037  0000000G               CMPB    R0,CORUSR       ;;;Do we already own buffer?
002344  001415                         BEQ     2$              ;;;Br if yes
                                ;
                                ;  Someone else owns the buffer now.
                                ;  Suspend our job until we can get it.
                                ;
002346                                  ENABL                   ;Enable interrupts
002354  012700  0000000G               MOV     #S$QCXB,R0      ;Get waiting-for-buffer wait state
002360  004737  0000000G               CALL    QNSPNX          ;Suspend job until buffer is available
002364  004737  0000000G               CALL    CHKABT          ;Were we aborted while asleep?
002370  000755                         BR      1$              ;Go try to get buffer
                                ;
                                ;  We can get buffer now
                                ;
002372  113737  0000000G 0000000G 3$:      MOVB    CORUSR,CXBOWN   ;;;Claim buffer for us
002400                          2$:     ENABL                   ;Enable interrupts
                                ;
                                ;  Finished
                                ;
002406  000207                          RETURN
                                        .SBTTL  FRECXT -- Free the context block buffer
                                ;-------------------------------------------------------------------
                                ;  This routine is called to release our ownership of the buffer
                                ;  used to access the job context block of another job (CXTBUF).
                                ;
002410  123737  0000000G 0000000G FRECXT: CMPB    CORUSR,CXBOWN   ;Do we own the buffer now?
002416  001010                         BNE     9$              ;Br if not
002420  105037  0000000G               CLRB    CXBOWN          ;Say the buffer is free
002424  105037  0000000G               CLRB    CXBJOB          ;Say no job's context data in buffer
                                ;
                                ;  Restart any jobs waiting for the buffer
                                ;
002430  012700  0000000G               MOV     #S$QCXB,R0      ;Get wait state
002434  004737  0000000G               CALL    UREGO           ;Restart waiting jobs
                                ;
                                ;  Finished
                                ;
002440  000207                  9$:     RETURN
```

```
                                        .SBTTL  GETCHA -- Get address of user's channel block
                                ;-------------------------------------------------------------------
                                ;  This routine is called to obtain a pointer to a specified I/O channel
                                ;  block for a specific job.
                                ;  Note:  GETCXT must have been called to gain exclusive access to the
                                ;         job context buffer before this routine can be used.
                                ;
                                ;  Inputs:
                                ;   R2 = Job index number
                                ;   R3 = Channel number
                                ;
                                ;  Outputs:
                                ;   R0 = Address of channel block
                                ;
002442  010346                  GETCHA: MOV     R3,-(SP)
                                ;
                                ;  Get the simulated RMON for the specified job into CXTBUF
                                ;
002444  004737  002510'                 CALL    REDRMN          ;Get simulated RMON into CXTBUF
                                ;
                                ;  Calculate address of specified channel block
                                ;
002450  013700  000000G                 MOV     CXTBUF,R0       ;Point to RMON data in buffer
002454  062700  000000G                 ADD     #R$CHN,R0       ;Point to 1st channel block
002460  020327  000021                  CMP     R3,#17.         ;Is this channel in extended area?
002464  103404                          BLO     1$              ;Br if not
002466  162703  000021                  SUB     #17.,R3         ;Get channel # within extended channel area
002472  062700  000000C                 ADD     #R$XCHN-R$CHN,R0;Point to extended channel area
002476  070327  000000G         1$:     MUL     #CHNSIZ,R3      ;Multiply by # bytes per channel block
002502  060300                          ADD     R3,R0           ;Get address of specified channel block
                                ;
                                ;  Finished
                                ;
002504  012603                          MOV     (SP)+,R3
002506  000207                          RETURN
```

```
                                        .SBTTL  REDRMN -- Access simulated RMON for a job
                                ;-----------------------------------------------------------------
                                ;  This routine is called to read into the job context block buffer
                                ;  the simulated RMON (including I/O channel blocks) for a job.
                                ;
                                ;  Inputs:
                                ;   R2 = Index number of job whose RMON is to be read.
                                ;
                                ;  Outputs:
                                ;   (CXTBUF) = Rmon data.
                                ;
002510  010346                 REDRMN: MOV     R3,-(SP)
                                ;
                                ;  Set up information about the simulated RMON
                                ;
002512  013703  0000000                 MOV     CXTRMN,R3       ;Get address of base of RMON with cxt blk
002516  012700  0000000                 MOV     #MVSIZ,R0       ;Get size of simulated RMON
                                ;
                                ;  Get simulated RMON into context block buffer
                                ;
002522  004737  002726'                 CALL    REDCXT          ;Get RMON into buffer
                                ;
                                ;  Finished
                                ;
002526  012603                          MOV     (SP)+,R3
002530  000207                          RETURN
```

```
                                        .SBTTL  GTUKBL -- Access region used for user key definitions
                                ;-------------------------------------------------------------------
                                ;  This routine is used to access a 512-byte page of information in
                                ;  the local named region used to store user key definitions.
                                ;
                                ;  The form of the EMT is:
                                ;       .BYTE   0,126
                                ;       .WORD   17
                                ;       .WORD   job_number
                                ;       .WORD   block_number
                                ;       .WORD   destination_address
                                ;
                                ;  The following error codes can be returned:
                                ;   0 = Job does not have any user-defined keys.
                                ;   1 = Job is swapped out of memory.
                                ;
002532                          GTUKBL:
                                ;
                                ;  Gain exclusive access to context buffer
                                ;
002532  004737  002324'                CALL    GETCXT          ;Gain access to context buffer
002536  103002                        BCC     1$              ;Br if got it
002540  004737  000000G               CALL    STOP            ;Job was aborted while waiting
                                ;
                                ;  Read KEYRCB cell into buffer
                                ;
002544  013702  000040G         1$:     MOV     EMTBLK+4,R2     ;Get number of job we are accessing
002550  012703  000000G                 MOV     #KEYRCB,R3      ;Get address of cell we want
002554  012700  000002                  MOV     #2,R0           ;Get # bytes we need
002560  004737  002726'                CALL    REDCXT          ;Copy data from jobs KEYRCB cell
                                ;
                                ;  See if job has a region assigned for key definitions
                                ;
002564  013705  000000G                 MOV     CXTBUF,R5       ;Get pointer to base of context buffer
002570  011503                        MOV     (R5),R3         ;Get value of KEYRCB
002572  001005                        BNE     2$              ;Br if job has some user-defined keys
                                ;
                                ;  Job has no user-defined keys
                                ;
002574  004737  002410'                CALL    FRECXT          ;Free context block buffer
002600  005000                        CLR     R0              ;Return error code 0
002602  000137  000000G               JMP     SETERR
                                ;
                                ;  Job has some user-defined keys.
                                ;  Get the region control block into context buffer.
                                ;
002606  012700  000000G         2$:     MOV     #RC$$SZ,R0      ;Get # bytes to access
002612  004737  002726'                CALL    REDCXT          ;Copy RCB into cxtbuf
                                ;
                                ;  See if job is currently in memory.
                                ;
002616  032762  000000G 000000G         BIT     #$INCOR,LSW(R2) ;Is job in memory now?
002624  001011                        BNE     3$              ;Br if yes
                                ;
                                ;  Job is currently swapped out of memory
                                ;
002626  016537  000000G 000000G         MOV     RC$BLK(R5),UR0  ;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
```

```
                                        .SBTTL  REDCXT -- Read a job context block into buffer
                                ;-------------------------------------------------------------------
                                ;  This routine is called to read a portion of a job context block,
                                ;  belonging to some job other than the current job, into the
                                ;  CXTBUF buffer.
                                ;  GETCXT must have been called prior to calling this routine to
                                ;  guarantee exclusive access to the context block buffer.
                                ;
                                ;  Inputs:
                                ;   R2 = Job index number of the job whose context block is to be read.
                                ;   R3 = Address of item in context block which is to be positioned
                                ;        at the base of the buffer.
                                ;   R0 = Number of bytes to be read (512 maximum).
                                ;
                                ;  Outputs:
                                ;   CXTBUF contains specified data.
                                ;
002726  010346                 REDCXT: MOV     R3,-(SP)
002730  010546                         MOV     R5,-(SP)
                                ;
                                ;  See if the requested data is already in the buffer.
                                ;
002732  120237  000000G                 CMPB    R2,CXBJOB       ;Is data for right job in buffer?
002736  001006                         BNE     1$              ;Br if not
002740  020337  000000G                 CMP     R3,CXBBAS       ;Is base item correct?
002744  001003                         BNE     1$              ;Br if not
002746  023700  000000G                 CMP     CXBSIZ,R0       ;Do we have enough data in buffer?
002752  103026                         BHIS    9$              ;Br if yes -- We are finished
                                ;
                                ;  We do not have the correct data in the buffer.
                                ;  Set up information about what we will be getting into buffer.
                                ;
002754  110237  000000G         1$:     MOVB    R2,CXBJOB       ;Remember which context block we are getting
002760  010337  000000G                 MOV     R3,CXBBAS       ;Remember base offset
002764  005200                         INC     R0              ;Force size to be word multiple
002766  042700  000001                 BIC     #1,R0
002772  010037  000000G                 MOV     R0,CXBSIZ       ;Remember amt of data being gotten
                                ;
                                ;  See if the job whose context data we want is currently in memory.
                                ;
002776  032762  000000G 000000G         BIT     #$INCOR,LSW(R2) ;Is job in memory now?
003004  001407                         BEQ     2$              ;Br if not
                                ;
                                ;  Job is in memory.
                                ;  Do a memory-to-memory move to get the data.
                                ;
003006  016205  000000G                 MOV     LCXPAR(R2),R5   ;Get base PAR mapping value
003012  162703  000000G                 SUB     #CXTBAS,R3      ;Convert address to offset within cxt blk
003016  004737  000000G                 CALL    CXBMOV          ;Move data to context block buffer
003022  000402                         BR      9$
                                ;
                                ;  Job is not now in memory.
                                ;  Read the context block data from the swap file.
                                ;
003024  004737  003036'         2$:     CALL    CXBSWP          ;Read data from swap file
                                ;
                                ;  Finished
```

REDCXT -- Read a job context block into buffer

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

```
                        .SBTTL  CXBSWP -- Read job context data from swap file
                ;-----------------------------------------------------------------
                ;  Read some job context data from the swap file into CXTBUF buffer.
                ;
                ;  Inputs:
                ;   R2, CXBJOB = Index number of job whose context data is being read.
                ;   R3, CXBBAS = Address within context block of item to be read into
                ;                the base of the context buffer.
                ;   R0, CXBSIZ = Amt of data to be read (512. maximum)
                ;
003036  010146  CXBSWP: MOV     R1,-(SP)
003040  010246         MOV     R2,-(SP)
003042  010346         MOV     R3,-(SP)
003044  010446         MOV     R4,-(SP)
                ;
                ;  Determine which block in swap file has start of data being read
                ;
003046  162703  0000000        SUB     #CXTBAS,R3      ;Convert address to offset within context
003052  010301         MOV     R3,R1           ;Get starting offset
003054  072127  177767 ASH     #-9.,R1         ;Determine starting block # within context
003060  066201  0000000        ADD     LSWPBK(R2),R1   ;Add base block # within swap file
                ;
                ;  Determine how much data to read this time
                ;
003064  042703  177000         BIC     #^C<777>,R3     ;Get starting offset within 1st block
003070  010302         MOV     R3,R2
003072  060002         ADD     R0,R2           ;Get offset beyond last byte we need
003074  020227  001000         CMP     R2,#512.        ;We cannot read more than 512 bytes
003100  101402         BLOS    1$              ;Br if don't need to truncate
003102  012702  001000         MOV     #512.,R2        ;Read 512 bytes this time
                ;
                ;  Read data from swap file
                ;
003106  006202  1$:    ASR     R2              ;Convert # bytes to # words to read
003110                 .READW  #USREMT,#USWPCH,CXTBUF,R2,R1
                ;
                ;  If data is not positioned at the start of the block, move it
                ;  down to the base of the buffer.
                ;
003144  005703         TST     R3              ;Is data at base of buffer now?
003146  001434         BEQ     9$              ;Br if yes -- Don't need to move
003150  006302         ASL     R2              ;Get back # bytes read
003152  160302         SUB     R3,R2           ;Get # bytes we need to move down
003154  010200         MOV     R2,R0           ;Get # bytes to move
003156  006200         ASR     R0              ;Get # words we need to move down
003160  013704  0000000        MOV     CXTBUF,R4       ;Point to base of buffer
003164  060403         ADD     R4,R3           ;Point to start of data we want
003166  012324  2$:    MOV     (R3)+,(R4)+     ;Move the data down
003170  077002         SOB     R0,2$           ;Loop till all moved
                ;
                ;  See if the requested data spans a block boundary.
                ;
003172  013703  0000000        MOV     CXBSIZ,R3       ;Get total amt of data requested
003176  160203         SUB     R2,R3           ;Get # left to be read
003200  003417         BLE     9$              ;Br if we have gotten all that was requested
                ;
                ;  Read in second block to get remainder of data.
```

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

```
                                        .SBTTL  .CSISPC & .CSIGEN
                                ;-------------------------------------------------------------------------------
                                ;  Command string interpreters.
                                ;
003252  112737  000001  000000G CSISPC: MOVB    #1,SPCFLG       ;REMEMBER THIS IS .CSISPC
003260  000420                         BR      CSICOM
003262  105037  000000G         CSIGEN: CLRB    SPCFLG          ;REMEMBER THIS IS .CSIGEN
                                ;
                                ;  Close user's channels 0-10
                                ;
003266  012705  000010                 MOV     #10,R5          ;GET LAST CHANNEL # TO CLOSE
003272                          1$:    .CLOSE  R5              ;CLOSE THE CHANNEL
003302  005305                         DEC     R5              ;MORE TO DO?
003304  002372                         BGE     1$              ;BR IF YES
                                ;  .CSIGEN returns address above user's handler space in R0.
003306  013737  000100G 000000G       MOV     EMTBLK+10,UR0   ;RETURN HANDLER SPACE ADDRESS IN R0
003314  042737  000001  000000G       BIC     #1,UR0          ;MAKE SURE ADDRESS IS EVEN
                                ;
                                ;  Save the user's stack pointer on entry in case we have to
                                ;  restore it because we have a command error.
                                ;
003322  106506                 CSICOM: MFPD    SP              ;Get user's stack pointer
003324  012637  000000G                MOV     (SP)+,CSIUSP    ;Save in job context block cell
                                ;
                                ;  Print prompt and get command line
                                ;  or move command from user's area to our area.
                                ;
003330  013703  000004G         CSIGCL: MOV     EMTBLK+4,R3     ;ADDRESS OF USER'S COMMAND STRING
003334  001417                         BEQ     2$              ;BR IF INPUT IS FROM THE TERMINAL
                                ;  Move command string from user's area to our internal buffer.
003336  010300                         MOV     R3,R0           ;CHECK IF ADDRESS IS LEGAL
003340  004737  000000G               CALL    VALADB
003344  012702  000000G               MOV     #CSIBUF,R2      ;PUT COMMAND HERE
003350  004737  000000G         1$:    CALL    GETUCH          ;GET CHAR FROM USER'S AREA
003354  020227  000120G               CMP     R2,#<CSIBUF+80.>;DON'T GO BEYOND END OF BUFFER
003360  103003                         BHIS    9$              ;BR IF COMMAND LINE TOO LONG
003362  110022                         MOVB    R0,(R2)+        ;MOVE TO OUR BUFFER
003364  001371                         BNE     1$              ;LOOP TILL ASCIZ NULL HIT
003366  000435                         BR      3$              ;GO PROCESS THE COMMAND
003370  105012                 9$:     CLRB    (R2)            ;STORE NULL AT END OF COMMAND
003372  000433                         BR      3$              ;GO PROCESS THE COMMAND
                                ;  Prompt for and accept a command from the terminal.
003374  005005                 2$:     CLR     R5              ;ASSUME WE DON'T NEED TO SUSPEND COMMAND FILES
003376  023727  000100G 000003         CMP     EMTBLK+10,#3    ;IS THIS A .GTLIN EMT?
003404  101010                         BHI     6$              ;BR IF NOT
003406  001004                         BNE     8$              ;BR IF NOT FORCING INPUT FROM TERMINAL
003410  013705  000000G               MOV     CFPNT,R5        ;SAVE CURRENT COMMAND FILE BUFFER POINTER
003414  005037  000000G               CLR     CFPNT           ;SUSPEND INPUT FROM COMMAND FILE
003420  013702  000006G         8$:    MOV     EMTBLK+6,R2     ;GET ADDRESS OF USER'S PROMPT STRING
003424  000405                         BR      7$
003426  012702  000000G         6$:    MOV     #USRUCA,R2      ;PUT * IN USER'S AREA FOR PROMPT
003432  012746  100052                 MOV     #100052,-(SP)
003436  106612                         MTPD    (R2)            ;MOVE TO USER'S AREA
003440  012703  000000G         7$:    MOV     #CSIBUF,R3      ;PUT STRING HERE
003444                                 .CALL   ASKLIN          ;PRINT PROMPT & ACCEPT LINE FROM TERMINAL
003452  005705                         TST     R5              ;DID WE SUSPEND INPUT FROM COMMAND FILES?
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  000010G 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 .GTLINE EMT, we are finished
     77                                ;
     78 003526  023727  0000100 000003         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 .GTLINE
```

```
                                ;
                                ;  At this point the command string has been accrued and is stored
                                ;  in asciz form in CSIBUF.
                                ;
                                ;  If we are doing a .CSISPC, zero user's file spec block.
003542  105737  000000G      CSISCN: TSTB    SPCFLG          ;DOING .CSISPC?
003546  001417                      BEQ     4$              ;BR IF NOT
003550  013705  000010G             MOV     EMTBLK+10,R5    ;GET ADDRESS OF FILE SPEC BUFFER
003554  042705  000001              BIC     #1,R5           ;MAKE ADDRESS EVEN
003560  010500                      MOV     R5,R0           ;MAKE SURE ADDRESS IS LEGAL
003562  062700  000116              ADD     #2*39.,R0
003566  004737  000000G             CALL    VALADW
003572  010504                      MOV     R5,R4
003574  012703  000047              MOV     #39.,R3         ;FILE SPEC IS 39 WORDS LONG
003600  005046                  1$: CLR     -(SP)           ;ZERO USER'S AREA
003602  106624                      MTPD    (R4)+
003604  077303                      SOB     R3,1$
                                ;
                                ;  The following registers are set up for the actual command scan:
                                ;   R1 = Pointer into CSIBUF.
                                ;   R4 = File number we are working on.
                                ;   R5 = Pointer into user's device spec buffer (.CSISPC only).
                                ;
                                ;  If command line contains an equal sign then we have output file specs.
                                ;  Otherwise first file is an input file (# 3).
                                ;
003606  105037  000000G          4$:     CLRB    CSIEQL          ;HAVEN'T SEEN EQUAL SIGN YET
003612  105037  000000G                  CLRB    SWTCNT          ;NO SWITCHES YET
003616  013737  000002' 000000G         MOV     R50DK,CSIDEV    ;SET DEFAULT DEVICE NAME
003624  012701  000000G                  MOV     #CSIBUF,R1      ;POINT TO COMMAND STRING
003630  010102                          MOV     R1,R2
003632  112200                  2$:     MOVB    (R2)+,R0        ;SCAN ACROSS COMMAND LOOKING FOR EQUAL SIGN
003634  001410                          BEQ     3$              ;BR IF END OF COMMAND HIT -- NO EQUAL SIGN FOUND
003636  120027  000075                  CMPB    R0,#'=          ;DOES COMMAND CONTAIN AN EQUAL SIGN?
003642  001403                          BEQ     5$              ;BR IF YES
003644  120027  000074                  CMPB    R0,#'<          ;LESS-THAN SIGN IS EQUIVALENT TO EQUAL-SIGN
003650  001370                          BNE     2$
                                ;  Command line contains an equal sign.  Output files come first.
003652  005004                  5$:     CLR     R4              ;FIRST FILE # IS 0
003654  000406                          BR      CSICHR
                                ;  Command line does not contain an equal sign.  Output files are omitted.
003656  012704  000003          3$:     MOV     #3,R4           ;FIRST FILE IS # 3
003662  062705  000036                  ADD     #36,R5          ;POINT TO 1ST INPUT FILE SPEC AREA
003666  105237  000000G                  INCB    CSIEQL          ;SAY WE ARE BEYOND EQUAL SIGN
```

```
                                        ;
                                        ;  Begin processing the next file specification.
                                        ;
003672  004737  004660'         CSICHR: CALL    CSIGET          ;GET NEXT CHAR FROM COMMAND LINE
003676  105700                  CSICH1: TSTB    R0              ;REACHED END OF COMMAND LINE?
003700  001002                          BNE     7$              ;BR IF NOT
003702  000137  004504'                 JMP     CSIFIN          ;FINISH OFF COMMAND
003706  120027  000075          7$:     CMPB    R0,#'=          ;REACHED EQUAL SIGN?
003712  001403                          BEQ     17$             ;BR IF YES
003714  120027  000074                  CMPB    R0,#'<          ;LESS-THAN SIGN IS EQUIVALENT TO EQUAL SIGN
003720  001023                          BNE     1$              ;BR IF NOT
                                        ;  "=" means start of input file specs.
003722  105737  0000000G        17$:    TSTB    CSIEQL          ;HAVE WE SEEN AN EQUAL SIGN BEFORE?
003726  001402                          BEQ     11$             ;BR IF NOT
003730  000137  004520'                 JMP     CSIILL          ;ERROR IF 2 EQUAL SIGNS
003734  105237  0000000G        11$:    INCB    CSIEQL          ;REMEMBER EQUAL SIGN SEEN
003740  012704  000003                  MOV     #3,R4           ;NEXT FILE # IS 3
003744  013705  0000100G                MOV     EMTBLK+10,R5    ;GET ADDRESS OF USER'S FILE SPEC AREA
003750  042705  000001                  BIC     #1,R5           ;MAKE EVEN
003754  062705  000036                  ADD     #36,R5          ;POINT TO SPACE FOR 1ST INPUT FILE SPEC
003760  013737  000002' 0000000G        MOV     R50DK,CSIDEV    ;RESET DEFAULT DEVICE TO "DK:"
003766  000741                          BR      CSICHR          ;CONTINUE SCAN
                                        ;  See if we have a null file spec.
003770  120027  000054          1$:     CMPB    R0,#',          ;NULL FILE SPEC?
003774  001002                          BNE     14$             ;BR IF NOT
003776  000137  004460'                 JMP     CSINXT          ;ADVANCE TO NEXT FILE
                                        ;  See if we have a leading switch specification.
004002  120027  000057          14$:    CMPB    R0,#'/          ;START OF SWITCH SPECIFICATION?
004006  001002                          BNE     15$             ;BR IF NOT
004010  000137  004336'                 JMP     CSISWT          ;GO PROCESS THE SWITCH
                                        ;
                                        ;  Accrue a file specification.
                                        ;
004014  005301                  15$:    DEC     R1              ;POINT TO 1ST CHARACTER OF NAME
004016  020427  000010                  CMP     R4,#8.          ;HAVE WE ALREADY GOTTEN MAX # FILES ALLOWED?
004022  101006                          BHI     13$             ;BR IF TOO MANY
004024  105737  0000000G                TSTB    CSIEQL          ;ARE WE GETTING INPUT OR OUTPUT FILES?
004030  001005                          BNE     2$              ;BR IF INPUT FILES
004032  020427  000003                  CMP     R4,#3           ;ONLY ALLOW 3 OUTPUT FILES
004036  103402                          BLO     2$              ;BR IF OK
004040  000137  004520'         13$:    JMP     CSIILL          ;ERROR
                                        ;  Get the default extension for the file name.
004044  010400                  2$:     MOV     R4,R0           ;GET FILE NUMBER
004046  020027  000002                  CMP     R0,#2           ;INPUT OR OUTPUT FILE?
004052  101402                          BLOS    12$             ;BR IF OUTPUT FILE
004054  005000                          CLR     R0              ;USE EXTENSION 0 FOR ALL INPUT FILES
004056  000402                          BR      16$
004060  005200                  12$:    INC     R0              ;CONVERT FILE # TO INDEX INTO EXTENSION TABLE
004062  006300                          ASL     R0              ;2 BYTES PER ENTRY
004064  063700  0000006G        16$:    ADD     EMTBLK+6,R0     ;ADD BASE OF DEFAULT EXT TABLE
004070  004737  0000000G                CALL    VALADW          ;MAKE SURE ADDRESS IS LEGAL
004074  106510                          MFPD    (R0)            ;GET DEFAULT EXTENSION FROM USER'S TABLE
004076  012600                          MOV     (SP)+,R0        ;GET DEFAULT EXTENSION
                                        ;  Parse the device-file specification.
004100  004737  004726'                 CALL    ACRFIL          ;ACCRUE THE FILE SPECIFICATION
004104  103002                          BCC     19$             ;BR IF NO ERROR
004106  000137  004520'                 JMP     CSIILL          ;INVALID SPEC
```

```
                                ;  Require a file name with file structured output devices.
004112  020427  000003      19$:    CMP     R4,#3           ;IS THIS AN OUTPUT FILE SPEC?
004116  103037                      BHIS    4$              ;BR IF NOT
004120                              .DSTATUS #CSIARE,#CSIFIL;GET INFO ABOUT THE DEVICE
004132  103020                      BCC     9$              ;BR IF VALID DEVICE
004134  105737  000000G             TSTB    SPCFLG          ;IS THIS .CSISPC OR .CSIGEN?
004140  001571                      BEQ     ILLDEV          ;BR IF .CSIGEN WITH INVALID DEVICE
                                ;
                                ;  .CSISPC:  Move file spec to user's area.
                                ;
004142  012700  000004      21$:    MOV     #4,R0           ;ASSUME 4 WORDS TO MOVE
004146  020427  000003              CMP     R4,#3           ;INPUT OR OUTPUT FILE SPEC?
004152  103001                      BHIS    5$              ;BR IF INPUT
004154  005200                      INC     R0              ;MOVE 5 WORDS FOR OUTPUT FILE SPECS
004156  012702  000000G     5$:     MOV     #CSIFIL,R2      ;POINT TO BUFFER WITH FILE SPEC WE GOT
004162  010503                      MOV     R5,R3           ;GET POINTER TO FILE SPEC AREA
004164  012246              6$:     MOV     (R2)+,-(SP)     ;MOVE A WORD FROM FILE SPEC TO USER'S AREA
004166  106623                      MTPD    (R3)+
004170  077003                      SOB     R0,6$           ;MOVE ALL OF SPEC
004172  000457                      BR      10$
                                ;
                                ;  .CSIGEN:  See if file specified or device non-file structured.
                                ;
004174  105737  000000G     9$:     TSTB    SPCFLG          ;.CSISPC?
004200  001360                      BNE     21$             ;IF SO, DON'T REQUIRE FILE NAME
004202  005737  000020G             TST     CSIFIL+2        ;WAS A FILE NAME SPECIFIED?
004206  001003                      BNE     4$              ;BR IF YES
004210  005737  000000G             TST     CSIARE          ;IS DEVICE FILE STRUCTURED?
004214  002541                      BLT     CSIILL          ;ERROR IF YES
                                ;
                                ;  .CSIGEN:  Do .lookup or .enter for file.
                                ;
004216  105737  000000G     4$:     TSTB    SPCFLG          ;CHECK FOR .CSISPC OR .CSIGEN
004222  001347                      BNE     21$             ;BR IF .CSISPC
004224  020427  000003              CMP     R4,#3           ;INPUT OR OUTPUT FILE?
004230  103025                      BHIS    8$              ;BR IF INPUT
                                ;  Output:  Enter the file.
004232                              .ENTER  #CSIARE,R4,#CSIFIL,CSIFIL+10
004262  103023                      BCC     10$             ;BR IF ENTER WAS SUCCESSFUL
004264  123727  000000G 000003     CMPB    @#ERRLOC,#3     ;PROTECTED FILE CONFLICT?
004272  001002                      BNE     18$             ;BR IF NOT
004274  000137  004532'             JMP     CSIPRO          ;RETURN ERROR CODE 2
004300  000137  004540'     18$:    JMP     CSIFUL          ;RETURN ERROR CODE 3
                                ;  Input:  Lookup the file.
004304              8$:             .LOOKUP #CSIARE,R4,#CSIFIL
004326  103001                      BCC     10$             ;BR IF OK
004330  000506                      BR      CSINOF          ;NO FILE FOUND
004332  000137  003672'     10$:    JMP     CSICHR          ;CONTINUE SCAN
```

```
                                        ;
                                        ;  Process a switch specification.
                                        ;
004336  004737  004660'       CSISWT: CALL    CSIGET          ;GET NEXT CHAR
004342  010003                       MOV     R0,R3           ;GET SWITCH CHARACTER
004344  001465                       BEQ     CSIILL          ;BR IF NO CHAR FOLLOWS "/"
004346  042703  177400         3$:   BIC     #^C<377>,R3     ;LEAVE ONLY 8-BITS
004352  010400                       MOV     R4,R0           ;GET FILE #
004354  000300                       SWAB    R0              ;PUT IN LEFT BYTE
004356  050003                       BIS     R0,R3           ;COMBINE FILE # WITH SWITCH CHARACTER
                                        ;  See if switch has an associated value.
004360  004737  004660'              CALL    CSIGET          ;GET NEXT CHAR
004364  120027  000072               CMPB    R0,#':          ;IS THERE AN ASSOCIATED VALUE?
004370  001023                       BNE     2$              ;BR IF NOT
                                        ;
                                        ;  Switch has a value -- Get it and push onto user's stack.
                                        ;
004372  052703  100000               BIS     #100000,R3      ;SET VALUE FLAG
004376  004737  005324'         1$:   CALL    SWVAL           ;GET VALUE
004402  103470                       BCS     BADVAL          ;BR IF INVALID
                                        ;  Push switch value onto user's stack.
004404  004737  005704'              CALL    UPUSH           ;PUSH ONTO USER'S STACK
004410  105237  000000G              INCB    SWTCNT          ;COUNT # SWITCHES ACCRUED
                                        ;  Push switch control word on user's stack.
004414  010300                       MOV     R3,R0           ;GET CONTROL WORD
004416  004737  005704'              CALL    UPUSH           ;PUSH ONTO USER'S STACK
                                        ;  See if there are additional values for this switch.
004422  004737  004660'              CALL    CSIGET          ;GET NEXT CHAR
004426  120027  000072               CMPB    R0,#':          ;IS THERE ANOTHER VALUE
004432  001761                       BEQ     1$              ;BR IF YES
004434  000137  003676'              JMP     CSICH1          ;CONTINUE SCAN
                                        ;
                                        ;  No switch value
                                        ;
004440  005301                 2$:   DEC     R1              ;BACK UP CHARACTER POINTER
004442  010300                       MOV     R3,R0           ;GET SWITCH CONTROL WORD
004444  004737  005704'              CALL    UPUSH           ;PUSH ONTO USER'S STACK
004450  105237  000000G              INCB    SWTCNT          ;COUNT # OF SWITCHES
004454  000137  003672'              JMP     CSICHR          ;CONTINUE SCAN
                                        ;
                                        ;  Finished a file spec.
                                        ;
004460  062705  000010       CSINXT: ADD     #10,R5          ;POINT TO NEXT FILE SPEC AREA
004464  020427  000003               CMP     R4,#3           ;DOING AN INPUT OR OUTPUT FILE?
004470  103002                       BHIS    1$              ;BR IF AN INPUT FILE
004472  062705  000002               ADD     #2,R5           ;SKIP 2 MORE BYTES
004476  005204                 1$:   INC     R4              ;ADVANCE FILE #
004500  000137  003672'              JMP     CSICHR          ;CONTINUE SCAN
                                        ;
                                        ;  Finished scanning command line.
                                        ;  Return number of switches on top of user's stack.
                                        ;
004504  113700  000000G       CSIFIN: MOVB    SWTCNT,R0       ;GET # SWITCHES ACCRUED
004510  004737  005704'              CALL    UPUSH           ;PUSH ONTO USER'S STACK
004514  000137  000000G              JMP     EMTXIT          ;FINISHED
```

```
                                        ;
                                        ;  CSI errors.
                                        ;
                                        ;  Illegal command
004520  005003                  CSIILL: CLR     R3              ;ERROR CODE = 0
004522  000424                          BR      CSIERR
                                        ;  Illegal device
004524  012703  000001          ILLDEV: MOV     #1,R3           ;ERROR CODE = 1
004530  000421                          BR      CSIERR
                                        ;  Enter failed -- Protected file conflict.
004532  012703  000002          CSIPRO: MOV     #2,R3           ;ERROR CODE = 2
004536  000416                          BR      CSIERR
                                        ;  Enter failed -- Insufficient disk space.
004540  012703  000003          CSIFUL: MOV     #3,R3           ;ERROR CODE = 3
004544  000413                          BR      CSIERR
                                        ;  Lookup failed
004546  012703  000004          CSINOF: MOV     #4,R3           ;ERROR CODE = 4
004552  000410                          BR      CSIERR
                                        ;  Invalid switch
004554  005003                  BADSWT: CLR     R3              ;ERROR CODE = 0
004556  012702  000005                  MOV     #5,R2           ;SET ERROR MESSAGE NUMBER
004562  000405                          BR      CSIER1
                                        ;  Invalid switch value
004564  005003                  BADVAL: CLR     R3              ;ERROR CODE = 0
004566  012702  000006                  MOV     #6,R2           ;ERROR MESSAGE NUMBER
004572  000401                          BR      CSIER1
                                        ;
                                        ;  Common CSI error processing.
                                        ;
004574  010302                  CSIERR: MOV     R3,R2           ;SET ERROR MESSAGE NUMBER
                                        ;  Restore initial user stack pointer
004576  013746  000000G         CSIER1: MOV     CSIUSP,-(SP)    ;Get initial stack pointer for user
004602  106606                          MTPD    SP              ;Reset for user
                                        ;  If input is coming from the terminal, print error message and
                                        ;  retry command.
004604  005737  000040G                 TST     EMTBLK+4        ;INPUT FROM THE TERMINAL?
004610  001020                          BNE     3$              ;BR IF NOT
004612                                  OCALL   CSIMSG          ;PRINT ERROR MESSAGE
                                        ;  If this is .CSIGEN, purge channels we opened.
004620  105737  000000G                 TSTB    SPCFLG          ;CSISPC OR CSIGEN?
004624  001010                          BNE     1$              ;BR IF CSISPC
004626  005704                          TST     R4              ;DID WE OPEN ANY CHANNELS?
004630  001406                          BEQ     1$              ;BR IF NOT
004632                          2$:     .PURGE  R4              ;PURGE A CHANNEL
004642  005304                          DEC     R4              ;MORE TO DO?
004644  002372                          BGE     2$              ;BR IF YES
004646  000137  003330'        1$:     JMP     CSIGCL          ;GO BACK AND RETRY COMMAND
                                        ;  Input is coming from a user supplied string.  Return an error code.
004652  010300                  3$:     MOV     R3,R0           ;GET ERROR CODE
004654  000137  000000G                 JMP     SETERR          ;ABORT EMT
```

```
                                        .SBTTL  *** Subroutines ***
                                        .SBTTL  CSIGET -- Get next character from CSI command line
                                ;-------------------------------------------------------------------
                                ;  CSIGET is called to get the next non-blank character from the CSI
                                ;  command line.  Lower case letters are converted to upper case.
                                ;
                                ;  Inputs:
                                ;   R1 = Pointer to character to be gotten
                                ;
                                ;  Outputs:
                                ;   R0 = Character acquired
                                ;   R1 = Pointer past character gottenn
                                ;   z-flag set if character = null
                                ;
004660  112100          CSIGET: MOVB    (R1)+,R0        ;GET NEXT CHARACTER
004662  001420                  BEQ     2$              ;BR IF HIT STRING DELIMITER
004664  120027  000040          CMPB    R0,#' '         ;IS CHAR A SPACE?
004670  001773                  BEQ     CSIGET          ;SKIP OVER SPACES
004672  120027  000009          CMPB    R0,#TAB         ;IS CHARACTER A TAB?
004676  001770                  BEQ     CSIGET          ;SKIP TABS
004700  120027  000141          CMPB    R0,#141         ;LOWER CASE A?
004704  103405                  BLO     1$              ;BR IF NOT LOWER CASE LETTER
004706  120027  000172          CMPB    R0,#172         ;LOWER CASE Z?
004712  101002                  BHI     1$              ;BR IF NOT LOWER CASE LETTER
004714  162700  000040          SUB     #40,R0          ;CONVERT LOWER CASE TO UPPER CASE
004720  042700  177600  1$:     BIC     #^C<177>,R0     ;RETURN ONLY 7-BIT CHAR
004724  000207          2$:     RETURN
```

```
                                        .SBTTL  ACRFIL -- Accrue a file specification
                                ;-------------------------------------------------------------------------
                                ;  ACRFIL is called to parse a file specification of the form
                                ;  dev:file.ext[size].
                                ;
                                ;  Inputs:
                                ;   R0 = Default extension
                                ;   R1 = Pointer to start of file spec
                                ;   CSIDEV = Default device (Rad50)
                                ;
                                ;  Outputs:
                                ;   R1 = Pointer to character past end of file spec
                                ;   CSIFIL = 5 word buffer containing Rad50 file spec accrued
                                ;   CSIDEV is updated if a device is explicitly specified
                                ;   C-flag is set on return if the file spec is invalid
                                ;
004726  010446                  ACRFIL: MOV     R4,-(SP)
004730  010546                          MOV     R5,-(SP)
                                ;
                                ;  Initialize CSIFIL buffer.
                                ;
004732  012705  0000000                 MOV     #CSIFIL,R5      ;POINT TO RESULT BUFFER
004736  013725  0000000                 MOV     CSIDEV,(R5)+    ;PUT IN DEFAULT DEVICE
004742  005025                  1$:     CLR     (R5)+           ;CLEAR REST OF BUFFER
004744  020527  0000100                 CMP     R5,#CSIFIL+8.
004750  101774                          BLOS    1$
004752  010037  0000060                 MOV     R0,CSIFIL+6     ;SET DEFAULT EXTENSION
                                ;
                                ;  First character of file spec must be valid Rad50 character or period.
                                ;
004756  004737  004660'                 CALL    CSIGET          ;GET 1ST CHAR OF FILE SPEC
004762  005301                          DEC     R1              ;BACKUP CHAR POINTER SO WE WILL GET CHAR AGAIN
004764  120027  000056                  CMPB    R0,#'.          ;PERIOD?
004770  001403                          BEQ     6$              ;BR IF YES
004772  004737  005222'                 CALL    R50CHR          ;VALID RAD50 CHARACTER?
004776  103453                          BCS     10$             ;BR IF INVALID
                                ;
                                ;  Get first field which will be device or file name.
                                ;
005000  004737  005136'         6$:     CALL    GTRD50          ;ACCRUE RAD50 VALUE
005004  121127  000072                  CMPB    (R1),#':        ;IS THIS A DEVICE OR FILE NAME?
005010  001012                          BNE     2$              ;BR IF FILE NAME
005012  005201                          INC     R1              ;POINT PAST COLON
005014  010437  0000000                 MOV     R4,CSIFIL       ;SET DEVICE NAME
005020  001442                          BEQ     10$             ;BR IF NULL DEVICE NAME
005022  010437  0000000                 MOV     R4,CSIDEV       ;SET NEW DEFAULT DEVICE NAME
005026  005705                          TST     R5              ;WAS DEVICE NAME LONGER THAN 3 CHARS?
005030  001036                          BNE     10$             ;ERROR IF YES
                                ;
                                ;  Accrue the file name.
                                ;
005032  004737  005136'                 CALL    GTRD50          ;ACCRUE NEXT RAD50 FIELD
005036  010437  0000020         2$:     MOV     R4,CSIFIL+2     ;FIRST 3 CHARS OF FILE NAME
005042  010537  0000040                 MOV     R5,CSIFIL+4     ;SECOND 3 CHARS OF FILE NAME
005046  121127  000056                  CMPB    (R1),#'.        ;IS AN EXTENSION SPECIFIED?
005052  001010                          BNE     3$              ;BR IF NOT
005054  005201                          INC     R1              ;POINT PAST PERIOD
```

```
                                        ;
                                        ;  Accrue the extension.
                                        ;
005056  004737  005136'                        CALL    GTRD50          ;ACCRUE THE EXTENSION FIELD
005062  010437  000060G                       MOV     R4,CSIFIL+6     ;STORE THE EXTENSION
005066  121127  000056                        CMPB    (R1),#'.        ;Was another period specified?
005072  001415                               BEQ     10$             ;Br if yes -- Error
                                        ;
                                        ;  See if file size was specified.
                                        ;
005074  121127  000133          3$:           CMPB    (R1),#'[        ;WAS FILE SIZE SPECIFIED?
005100  001010                               BNE     4$              ;BR IF NOT
005102  005201                               INC     R1              ;POINT PAST OPEN BRACKET
                                        ;  Get the file size
005104  004737  005476'                       CALL    ACRDEC          ;ACCRUE A DECIMAL VALUE
005110  010037  000010G                       MOV     R0,CSIFIL+8.    ;STORE FILE SIZE
005114  122127  000135                        CMPB    (R1)+,#']       ;GOT PROPER DELIMITER?
005120  001002                               BNE     10$             ;ERROR IF NOT
                                        ;
                                        ;  Successful return
                                        ;
005122  000241                  4$:           CLC                     ;SIGNAL GOOD RETURN
005124  000401                               BR      5$
                                        ;
                                        ;  Error return
                                        ;
005126  000261                  10$:          SEC                     ;SIGNAL ERROR RETURN
005130  012605                  5$:           MOV     (SP)+,R5
005132  012604                               MOV     (SP)+,R4
005134  000207                               RETURN
```

```
                                        .SBTTL  GTRD50 -- Accrue a RAD50 name
                                ;-------------------------------------------------------------------------
                                ;  GTRD50 is called to accrue a name and return it as a Rad50 value.
                                ;
                                ;  Inputs:
                                ;   R1 = Pointer to start of name
                                ;   SPCFLG: 1==> "*" and "%" legal in name; 0==> not legal.
                                ;
                                ;  Outputs
                                ;   R1 = Pointer left pointing to name delimiter
                                ;   R4 = First 3 characters of name in Rad50
                                ;   R5 = Second 3 characters of name
                                ;
005136  012746  022000          GTRD50: MOV     #22000,-(SP)    ;COUNTER TO CONTROL # CHARS ACQUIRED
005142  005005                  4$:     CLR     R5              ;ACCRUE VALUE IN R5
                                ;  Get next character and see if it is valid for Rad50 name.
005144  004737  004660'          1$:     CALL    CSIGET          ;GET NEXT CHARACTER
005150  004737  005222'                  CALL    R50CHR          ;IS THIS A VALID RAD50 CHAR?
005154  103001                          BCC     2$              ;BR IF VALID
005156  005301                          DEC     R1              ;BACK UP CHAR POINTER TO DELIMITER
                                ;  We got a valid Rad50 character.
                                ;  Add to accumulated value.
005160  070527  000050          2$:     MUL     #50,R5          ;MULTIPLY BY 50
005164  060005                          ADD     R0,R5           ;ADD IN NEW CHARACTER
005166  006316                          ASL     (SP)            ;HAVE WE GOTTEN 3 CHARS YET?
005170  103365                          BCC     1$              ;BR IF NOT
005172  005716                          TST     (SP)            ;HAVE WE GOTTEN 6 CHARS?
005174  001402                          BEQ     3$              ;BR IF YES
005176  010504                          MOV     R5,R4           ;RETURN 1ST 3 CHARS IN R4
005200  000760                          BR      4$              ;GO GET NEXT 3 CHARS
                                ;  Skip any chars in name after sixth.
005202  004737  004660'          3$:     CALL    CSIGET          ;GET NEXT CHAR
005206  004737  005222'                  CALL    R50CHR          ;IS THIS A DELIMITER?
005212  103373                          BCC     3$              ;SKIP IT IF NOT
005214  005301                          DEC     R1              ;POINT BACK TO DELIMITER
005216  005726                          TST     (SP)+           ;REMOVE COUNT FROM STACK
005220  000207                          RETURN
```

```
                                        .SBTTL  R50CHR -- Convert ascii character to Rad50 value
                                ;-------------------------------------------------------------------------
                                ;  R50CHR is called to convert an ascii character to its equivalent
                                ;  Rad50 value.
                                ;
                                ;  Inputs
                                ;   R0 = Ascii character
                                ;   SPCFLG:  1==> allow "*" and "%"; 0==> do not allow * and %
                                ;
                                ;  Outputs
                                ;   R0 = Rad50 value of character
                                ;   C-flag set if character is not a valid Rad50 character (value returned = 0)
                                ;
005222  010146                  R50CHR: MOV     R1,-(SP)
005224  012701  005302'                 MOV     #R50RNG,R1      ;POINT TO CONVERSION TABLE
                                ;  Search for ascii char in our conversion table
005230  120011                  1$:     CMPB    R0,(R1)         ;COMPARE WITH LOW VALUE IN RANGE
005232  103413                          BLO     2$              ;BR IF TOO SMALL FOR RANGE
005234  120061  000001                  CMPB    R0,1(R1)        ;COMPARE WITH HIGH VALUE IN RANGE
005240  101010                          BHI     2$              ;BR IF TOO LARGE FOR RANGE
005242  122121                          CMPB    (R1)+,(R1)+     ;IS THIS A SPECIAL CHAR (* OR %)?
005244  001003                          BNE     3$              ;BR IF NOT
005246  105737  0000000               TSTB    SPCFLG          ;SHOULD WE ALLOW SPECIAL CHARS?
005252  001407                          BEQ     9$              ;BR IF NOT
                                ;  We have a valid character.  Convert ascii value to rad50.
005254  061100                  3$:     ADD     (R1),R0         ;CONVERT ASCII VALUE TO RAD50 VALUE
005256  000241                          CLC                     ;SIGNAL GOOD RETURN
005260  000406                          BR      10$             ;GO RETURN
                                ;  Try next range in table.
005262  062701  000004          2$:     ADD     #4,R1           ;POINT TO NEXT ENTRY IN CONVERSION TABLE
005266  105711                          TSTB    (R1)            ;HAVE WE HIT END OF TABLE?
005270  001357                          BNE     1$              ;BR IF NOT
                                ;  Invalid character
005272  005000                  9$:     CLR     R0              ;RETURN 0 AS VALUE
005274  000261                          SEC                     ;SIGNAL ERROR
                                ;  Return
005276  012601                  10$:    MOV     (SP)+,R1
005300  000207                          RETURN
                                ;
                                ;  Rad50 conversion table.
                                ;  Values:  Low-value, high-value, Rad50-Ascii-value
                                ;
005302      101     132         R50RNG: .BYTE   'A,'Z           ;A-Z
005304  177700                          .WORD   1-'A
005306      060     071                 .BYTE   '0,'9           ;0-9
005310  177756                          .WORD   36-'0
005312      052     052                 .BYTE   '*,'*           ;*
005314  177763                          .WORD   -15
005316      045     045                 .BYTE   '%,'%           ;%
005320  177767                          .WORD   -11
005322  000000                          .WORD   0               ;END OF TABLE MARKER
                                        .EVEN
```

```
                                        .SBTTL  SWVAL  -- Accrue CSI switch value
                                ;-----------------------------------------------------------------
                                ;  SWVAL is called to accrue a value associated with a CSI switch.
                                ;  The value may be specified as Rad50 alphanumeric, decimal or octal.
                                ;
                                ;  Inputs:
                                ;   R1 = Pointer to start of value.
                                ;
                                ;  Outputs:
                                ;   R0 = Value accrued.
                                ;   R1 = Pointer to value delimiter.
                                ;   C-flag set on return if an error detected.
                                ;
005324  010546                  SWVAL:  MOV     R5,-(SP)
005326  010105                          MOV     R1,R5           ;SAVE POINTER TO START OF VALUE
                                ;
                                ;  Determine what type of value this is.
                                ;
005330  004737  004660'                 CALL    CSIGET          ;GET FIRST CHAR OF VALUE
005334  120027  000101                  CMPB    R0,#'A          ;ALPHABETIC?
005340  103414                          BLO     1$              ;BR IF NOT
005342  120027  000132                  CMPB    R0,#'Z
005346  101050                          BHI     5$              ;BR IF NOT ALPHABETIC
                                ;
                                ;  Rad50 value.
                                ;
005350  010501                          MOV     R5,R1           ;GET BACK POINTER TO START OF VALUE
005352  010446                          MOV     R4,-(SP)
005354  004737  005136'                 CALL    GTRD50          ;ACCRUE A RAD50 VALUE
005360  010400                          MOV     R4,R0           ;GET 1ST 3 CHARS OF VALUE
005362  012604                          MOV     (SP)+,R4
005364  005705                          TST     R5              ;2ND 3 CHARS SHOULD BE NULL
005366  001436                          BEQ     10$             ;BR IF OK
005370  000437                          BR      5$              ;VALUE TOO LONG
                                ;
                                ;  Value is not Rad50.
                                ;  Make sure it is numeric.
                                ;
005372  120027  000055          1$:     CMPB    R0,#'-          ;ALLOW LEADING MINUS SIGN
005376  001406                          BEQ     2$
005400  120027  000060                  CMPB    R0,#'0          ;IS VALUE NUMERIC?
005404  103431                          BLO     5$              ;BR IF NOT
005406  120027  000071                  CMPB    R0,#'9
005412  101026                          BHI     5$              ;BR IF NOT NUMERIC
                                ;
                                ;  Value is numeric.
                                ;  See if a decimal point was specified with value.
                                ;
005414  004737  004660'          2$:     CALL    CSIGET          ;GET NEXT CHAR IN VALUE
005420  120027  000056                  CMPB    R0,#'.          ;DECIMAL POINT?
005424  001413                          BEQ     4$              ;BR IF YES
005426  120027  000060                  CMPB    R0,#'0          ;ARE WE STILL PASSING DIGITS?
005432  103403                          BLO     3$              ;BR IF NOT
005434  120027  000071                  CMPB    R0,#'9
005440  101765                          BLOS    2$              ;KEEP LOOKING FOR DEC PT
                                ;
                                ;  Octal value.
```

```
                        ;
005442  010501          3$:     MOV     R5,R1           ;GET POINTER TO START OF VALUE
005444  004737  005572'          CALL    ACROCT          ;ACCRUE AN OCTAL VALUE
005450  103407                  BCS     5$              ;BR IF INVALID
005452  000404                  BR      10$
                        ;
                        ;  Decimal value.
                        ;
005454  010501          4$:     MOV     R5,R1           ;GET POINTER TO START OF VALUE
005456  004737  005476'          CALL    ACRDEC          ;ACCRUE A DECIMAL VALUE
005462  005201                  INC     R1              ;POINT PAST DECIMAL POINT
                        ;
                        ;  Successful finish
                        ;
005464  000241          10$:    CLC                     ;SIGNAL GOOD RETURN
005466  000401                  BR      11$
                        ;
                        ;  Error
                        ;
005470  000261          5$:     SEC                     ;SIGNAL ERROR ON RETURN
                        ;
                        ;  Return
                        ;
005472  012605          11$:    MOV     (SP)+,R5
005474  000207                  RETURN
```

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

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

```
      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
```

Cross reference table (CREF V05.04)

```
$DILUP    1-40      5-25
$INCOR    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
```

Cross reference table (CREF V05.04)

```
        34-16   34-20   35-17   35-21
CSIILL  26-15   26-41   26-57   26-86   27-6    28-5#
CSIMSG   1-44   28-38
CSINOF  26-104  28-17#
CSINXT  26-26   27-43#
CSIPRO  26-99   28-11#
CSISCN  24-64   24-79   25-6#
CSISPC   1-30   24-5#
CSISWT  26-30   27-4#
CSIUSP   1-44   24-23*  28-32
CXBBAS   1-49   22-25   22-34*
CXBJOB   1-49   18-47*  22-23   22-33*
CXBMOV   1-50   21-69   22-49
CXBOWN   1-49   18-17   18-33*  18-44   18-46*
CXBSIZ   1-49   22-27   22-37*  23-53
CXBSWP  22-55   23-11#
CXTBAS   1-51   22-48   23-18
CXTBUF   1-49    6-41    9-21   19-23   21-34   21-73   23-35   23-46
CXTRMN   1-51   10-15   20-16
CXXADR   6-32    6-54#   6-57
CXXSIZ   6-33    6-61#
DIEARG   1-71
DIEMSG   1-71
EMTBLK   1-42    2-5     2-10    3-15    3-21    3-25    3-27*   3-31    3-40    3-55    3-86    3-90
         4-24    4-25    4-26    5-20    5-41    5-51    6-26    6-39   11-19   12-9    12-31   13-9
        14-7*   14-29   15-20   21-27   21-66   21-74   24-16   24-28   24-44   24-49   24-63   24-66
        24-78   25-8    26-18   26-50   28-36
EMTPS    1-46   14-25
EMTXIT   1-43    2-21   3-104   4-32    5-59    6-50   12-45   13-26   14-52   21-83   24-80   27-55
ERRLOC   1-44   26-97
FIXPRV   3-69    3-96   3-110#
FRECXT   1-30    4-31    6-49    7-64    8-46   18-44#  21-40   21-58   21-82
GETCHA   1-31   19-15#
GETCXT   1-30    4-18    6-20    7-28    8-14   18-12#  21-21
GETQ     1-47   15-15
GETUCH   1-42   24-34
GTRD50  30-40   30-52   30-61   31-14#  33-29
GTUKBL   1-30   21-17#
ILLDEV  26-64   28-8#
INTPRI   1-51   18-25   18-34
JM$JOB   1-45   12-30*  15-21   17-25
JM$LNK   1-45   12-26   12-37*  12-40*  14-39   14-47   17-27   17-28*  17-32
JM$RTN   1-45    1-47   12-31*  15-23
JPWDEV   1-37    6-54
JPWFLG   1-37    6-56
JPWTYP   1-37    6-55
KEYRCB   1-36   21-28
KMCPCX   1-31    4-14#
LBSPRI   1-41    2-16*
LCXPAR   1-37   22-47
LDBASE   1-39    7-57
LDDEVX   1-36   10-30
LDFLAG   1-39    7-59
LDNAME   1-39    7-49
LDPDEV   1-39    7-52
LDSIZE   1-39    7-55
```

Cross reference table (CREF V05.04)

```
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
MON0     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#
MONFQH    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$$SZ    1-39      7-44
OKFILE    1-38      7-43
OVRHC     1-43      2-20      7-68     24-55     28-38
PO$SPV    1-41      3-23      3-78
PO$SYS    1-40     14-5
P2$CXT    1-40      5-34
P2$VIR    1-40      3-117
PR7       1-51
PRGLDC    7-18     10-5#
PRIVAO    1-41      3-45      3-65*     3-81      3-92*     8-20
PRIVCO    1-41      3-23      3-39      3-58*     3-78      3-85*     8-33*     14-5
PRIVC2    1-40      3-117     5-34
PRIVFO    1-35      3-62*     3-89*     8-34*
PRIVSO    1-41      3-42      3-61*     3-88*     8-26      8-33      8-34
PS        1-22#    18-16*    18-25*    18-34*
PUTUCH    1-43     24-72     24-74
PVNPW     1-41      3-46      3-67      3-94      8-21      8-31
QCOMPL    1-48     15-34
QNSPNX    1-50     18-27
R$CHN     1-52     10-16     10-20     19-24     19-28
R$XCHN    1-52     10-20     19-28
R50CHR   30-35     31-18     31-33     32-14#
R50DK     1-94#    25-29     26-21
R50RNG   32-15     32-43#
RC$$SZ    1-36     21-47
RC$BAS    1-37     21-65
```

Cross reference table (CREF V05.04)

```
RC$BLK    1-37     21-57
REDCXT    1-31      6-35     9-17     20-21     21-30     21-48     22-18#
REDRMN   19-19     20-12#
RESDEV    1-38      7-40
S$HICP    1-48     15-29
S$IOFN    1-48     15-25
S$QCXB    1-50     18-26    18-51
SETERR    1-44      1-46     3-100     5-28      5-37      6-31     12-25     12-50     13-31     14-17     21-42     21-60
         28-50
SETPRI    1-31      2-5#
SETPRV    2-6       3-11#
SMONHD    1-35     12-37    12-38*    14-14     14-44     17-16
SPCFLG    1-42     24-5*    24-7*     25-6      26-63     26-81     26-90     28-40     32-23
SPPRED    1-35      2-20
STOP      1-38      4-20     6-22      7-30      8-16     21-23
SWTCNT    1-43     25-28*   27-23*    27-38*    27-53
SWVAL    27-19     33-14#
SYSHLT    1-71
TAB       1-44     29-19
TSEM2     1-21#
UMODE     1-46     14-25
UPUSH    27-22     27-26    27-37     27-54     36-12#
URO       1-42     21-57*   24-16*    24-17*
UREGO     1-50     18-52
USREMT    1-51     23-35    23-63
USRUCA    1-42     24-51
USWPCH    1-49     23-35    23-35     23-63     23-63
VALADB    1-43     24-32    24-68
VALADW    1-42      3-17     6-40     25-12     26-51
VPRIDF    1-41      2-12
```

Cross reference table (CREF V05.04)

```
...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
```