Table of contents

Table of contents

```
                                        .TITLE  TSTTY -- TSX Terminal I/O routines
                                        .ENABL  LC
                                        .ENABL  AMA
                                        .DSABL  GBL
000000                                  .CSECT  TSTTY
000000  100071                  TSTTY:  .RAD50  /TTY/            ;Overlay region id
                                ;
                                ;  TSTTY is the TSX module that contains routines related
                                ;  to doing I/O to the user's terminal.
                                ;
                                ;  Copyright (c) 1980, 1981, 1982, 1983, 1984, 1985, 1986, 1987.
                                ;  S&H Computer Systems, Inc.  Nashville, Tn.
                                ;
                                ;
                                ;  Macro calls
                                ;
                                        .MCALL  .READW,.PURGE,.REOPEN,.WRITW
                                ;
                                ;  Global definitions
                                ;
                                        .GLOBL  PUTCHR,BUFCHR
                                        .GLOBL  DELCHR,CVTLC
                                        .GLOBL  TTINCP,TSTTY,LOGCHR,LOGCR
                                        .GLOBL  STRACT,PCSPND
                                        .GLOBL  WRITTT,TTREAD,CSIMSG,ASKLIN
                                        .GLOBL  PRINT,TTYOUT,TTYIN,SIGWAT,QUECHR,BUFCHR
                                        .GLOBL  XHISET,XHIOUT,XHIIN,XTERCK,XRDTIM
                                        .GLOBL  SETRBF,CMDB,CMDC,GTSPAC,CMDE,CMDF,CMDG,CMDH,CMDI
                                        .GLOBL  CMDJ,CMDK,CMDL,CMDM,CMDN,CMDO,RSSPAC,SFWAC,CMDR
                                        .GLOBL  CMDS,CMDT,CMDU,SFWL,CMDW,CMDX,CMDY,CMDZ,MAXCC
                                ;
                                ;  Global references
                                ;
                                        .GLOBL  $AUTO,$RBRK,$RFRSH,LSW4,R$CFST,CFACFL,$SCCA,AF$CCA,AFCF
                                        .GLOBL  SILFET,$SUSPN,WINPRT,$VBELL,LTTCR,TTCPL,AF$NPW,SUCF2
                                        .GLOBL  $RTCS,AUTSPD,DOSWIT,VVPWCH,PRIVFO,PRIVSO,PRIVCO,PVNPW
                                        .GLOBL  MXSPAC,LOTSIZ,LSTACT,SETERR,LSW11,$PWKEY,$NOWIN
                                        .GLOBL  TRNSTR,GTSLCH,$SLON,$SLTTY,$V52EM
                                        .GLOBL  $NOWTT,NOWAIT,FRKPRI,LSW7
                                        .GLOBL  CFARG,DISSLE,$DBKMN,$CTRLD,CXTRMN
                                        .GLOBL  LSW,LSW2,LSW3,LSW4,LSW5,LJSW,$NTGCC
                                        .GLOBL  $DILUP,$DOOFF,$DISCN,IN$ACT
                                        .GLOBL  $DETCH,$CTRLC,$1ESC,$NDICP
                                        .GLOBL  $SCOPE,$ECHO,CFHOLD,ILSW2,$CHACT
                                        .GLOBL  $TAB,$FORM,$NOINT,LABTIM
                                        .GLOBL  $LC,$NOVLN,$DEFER,$NOOUT,NEDCDI
                                        .GLOBL  $NOIN,$TRNSP,LINSPC,QCOMPL
                                        .GLOBL  $CTRLO,$RBOUT,$1STCH,$CTRLW
                                        .GLOBL  $CTRLS,$DODFR,$GCECO,$GCESC
                                        .GLOBL  $QUIET,$INKMN,$UCTLC,$SETCC,VVLSCH
                                        .GLOBL  $ODTMD,$CFOPN,$CFALL,$GTLIN
                                        .GLOBL  LSW10,$8BIT,LWINDO,WINCHR
                                        .GLOBL  LESCHR,LESRTN,LSNDCH
                                        .GLOBL  $TTERR,$CFSOT,$HITTY,$FLAGC
                                        .GLOBL  $1CTLC,$VTESC,HAZEL
                                        .GLOBL  S$$RUN,$DEBUG,$DBGBK
                                        .GLOBL  $DEAD
```

```
                        .GLOBL  BKSPAC,CR,$UKMON
                        .GLOBL  $LOFCF,$SUCF,LSW9
                        .GLOBL  LNMAP,S$OTFN,ENQTL,STOP
                        .GLOBL  CORUSR,CFPNT,LACTIV,LINPNT
                        .GLOBL  LINEND,LINBUF,MXCPRM,PRMPNT,LSCCA
                        .GLOBL  CURPRM,CFEND,CFCHAN,CFBUF,$SLINI
                        .GLOBL  LNSPAC,LSPACT,CTRLZ,CTRLC,CTRLX
                        .GLOBL  LINNXT
                        .GLOBL  TAB,BKSPAC,RUBOUT,FORCEX,LINSIZ,LINCNT
                        .GLOBL  BELL,LNPRIM,LF
                        .GLOBL  TRNSFL,ESC,LTSCMD,VTSLCH,LCBIT
                        .GLOBL  CFSPND,PR7,ACFLAG,LAFSIZ,INTPRI
                        .GLOBL  INTPRI,PSW,DOSCHD,ESCFLG
                        .GLOBL  INITFL,JSWLOC,STPFLG
                        .GLOBL  LCOL,LOTSPC,LOTNXT,LOTEND,LOTBUF
                        .GLOBL  S$OTWT,CHKABT
                        .GLOBL  QNSPNX,QHDSPN
                        .GLOBL  LOTPNT,FF
                        .GLOBL  LSTATE,SPCTTY,$CCLRN,CFBLK,LINCUR
                        .GLOBL  S$INWT
                        .GLOBL  MAXSEC,LRBFIL,SPACE
                        .GLOBL  LCBIT,INTPRI
                        .GLOBL  LRDTIM,LRTCHR,LBRKCH
                        .GLOBL  S$TTFN,VQUAN1,$NOLF,S$TTSC
                        .GLOBL  LBRKCQ,$DBGMD
                        .GLOBL  LFWLIM
                        .GLOBL  VT52,VT100,LTRMTP,VT2007,VT2008
                        .GLOBL  UHIMEM,TTCSCH
                        .GLOBL  $TAPE,$XSTOP,LSW6,KPAR6,LTTPAR,$CFABT
                        .GLOBL  BRKPT,$CFDCC,$CFCCL,$CFKIL
                        .GLOBL  GETUCH,PUTUCH,VALADB,EMTBLK,FAKCMP
                        .GLOBL  CS$EOF,CFLAG,CTRLZ,$FORMO,FF,OVRHC
                        .GLOBL  CHNADR,EMTPS,URO,GTLTTY,LJSW,SETC,EMTXIT
                        .GLOBL  CFIND,R$INST,CFNEST,CFSP,LSTPRM
                        .GLOBL  PBFEND,PRMBUF,CFLFL4,INITLN
                        .GLOBL  VINTIO,LHIPCT
                        .GLOBL  LOGBUF,LOGEND,LOGCHN,LOGBLK,ABORT,EMTADR
                        .GLOBL  LOGPTR,LOGFLG,LF$WRT,LITIME,LF$IN,LF$OUT
                ;
                ;   Macro definitions:
                ;
                        .MACRO  DISABL          ;DISABLE INTERRUPTS
                        BIS     #PR7,@#PSW
                        .ENDM   DISABL

                        .MACRO  ENABL           ;ENABLE INTERRUPTS
                        BIC     INTPRI,@#PSW
                        .ENDM   ENABL

                        .MACRO  OCALL   ENTADD
                        .IF     B,ENTADD
                                .ERROR  ;OCALL without entry address
                        .ENDC
                        CALL    OVRHC
                        .WORD   ENTADD
                        .ENDM   OCALL
                ;
```

```
                        ;  The TTMAP and TTMAPX macros are used to map kernel-mode par6 to the
                        ;  terminal character buffer area.  The previous contents of par6 map
                        ;  register are pushed on the stack and may be restored by using the
                        ;  UNMAP or UNMAPX macros.
                        ;  R1 must contain the line index number of the line whose buffers
                        ;  are being accessed.
                        ;  The difference between the TTMAP-UNMAP macros and the TTMAPX-UNMAPX
                        ;  macros is that the X-versions are more efficient but may only be
                        ;  used from within interrupt service routines where we are guaranteed
                        ;  to be running on the system stack.
                        ;  The TTMAP and UNMAP versions of the macros must only be
                        ;  used in sections of code where the interrupts are disabled.
                        ;
                                .MACRO  TTMAPX
                                MOV     LTTPAR(R1),@#KPAR6
                                .ENDM   TTMAPX

                        ;
                        ;
                                .MACRO  UNMAPX
                                .ENDM   UNMAPX
                        ;
                                .MACRO  TTMAP
                                MOV     @#KPAR6,MAPHLD
                                MOV     LTTPAR(R1),@#KPAR6
                                .ENDM   TTMAP
                        ;
                                .MACRO  UNMAP
                                MOV     MAPHLD,@#KPAR6
                                .ENDM   UNMAP
                        ;
                        ;  Data areas
                        ;
000002  000000          MAPHLD: .WORD   0               ;TEMP CELL USED BY TTMAP MACRO
                        ;
                        ;  CSI table of error messages.
                        ;
000004  000022'         CSIERR: .WORD   CSEMIL
000006  000051'                 .WORD   CSEMID
000010  000077'                 .WORD   CSEPRO
000012  000163'                 .WORD   CSEMFO
000014  000226'                 .WORD   CSEMNF
000016  000256'                 .WORD   CSEMIS
000020  000304'                 .WORD   CSEMIV
                        ;
                        ;  CSI text messages.
                        ;
                                .NLIST  BEX
000022      077     103     123 CSEMIL: .ASCIZ  /?CSI-F-Illegal command/
000051      077     103     123 CSEMID: .ASCIZ  /?CSI-F-Illegal device/
000077      077     103     123 CSEPRO: .ASCIZ  /?CSI-F-Protected file with same name already exists/
000163      077     103     123 CSEMFO: .ASCIZ  /?CSI-F-Insufficient space for file/
000226      077     103     123 CSEMNF: .ASCIZ  /?CSI-F-Cannot find file/
000256      077     103     123 CSEMIS: .ASCIZ  /?CSI-F-Invalid switch/
000304      077     103     123 CSEMIV: .ASCIZ  /?CSI-F-Invalid switch value/
                                .LIST   BEX
                        ;
```

```
                                        ;  Bit mask table used to test, set, and clear the activation-character
                                        ;  flags for characters.
                                        ;
000340      001     002     004  BITMSK:  .BYTE    1,2,4,10,20,40,100,200
000343      010     020     040
000346      100     200
                                                .EVEN
```

```
                                        .SBTTL  WRITTT -- .WRITE emt with terminal as output device
                                ;-------------------------------------------------------------
                                ;  WRITTT is executed when it is determined that the .WRITE emt is
                                ;  directed to the terminal device.
                                ;
000350  013703  000004G        WRITTT: MOV     EMTBLK+4,R3     ;GET BUFFER ADDRESS
000354  010300                  MOV     R3,R0           ;VALIDATE BUFFER ADDRESS
000356  004737  000000G          CALL    VALADB
000362  013704  000006G          MOV     EMTBLK+6,R4     ;GET # WORDS TO WRITE
000366  001454                  BEQ     2$              ;BR IF NO WORDS TO BE WRITTEN
000370  006304                  ASL     R4              ;CONVERT TO # BYTES
000372  060400                  ADD     R4,R0           ;GET ADDRESS OF END OF BUFFER
000374  005300                  DEC     R0
000376  004737  000000G          CALL    VALADB          ;VALIDATE IT
000402  042777  000000G 000000G  BIC     #CS$EOF,@CHNADR ;RESET CHANNEL END OF FILE FLAG
000410  005737  000002G          TST     EMTBLK+2        ;WRITE TO BLOCK 0?
000414  001010                  BNE     16$             ;BR IF NOT
000416  032761  000000G 000000G  BIT     #$FORMO,LSW4(R1);DOES HE WANT FF ON WRITE OF BLK 0?
000424  001404                  BEQ     16$             ;BR IF NOT
000426  112700  000000G          MOVB    #FF,R0          ;OUTPUT FF TO GET TO TOP OF FORM
000432  004737  002662'          CALL    PUTCHR
                                ;
                                ;  Determine if buffer is on even byte boundary
                                ;
000436  032703  000001          16$:    BIT     #1,R3           ;IS BUFFER STARTING ON EVEN BYTE BOUNDARY?
000442  001430                  BEQ     9$              ;BR IF YES
                                ;
                                ;  Use slow routine if buffer is on odd byte boundary
                                ;
000444  012702  000000G          17$:    MOV     #TTCSCH,R2      ;RESET CHARACTER COUNT
                                ;  See if we need to interrupt tt output processing to do a job
                                ;  scheduler cycle.
000450  105737  000000G                  TSTB    DOSCHD          ;IS JOB SCHEDULER CYCLE NEEDED?
000454  001410                  BEQ     4$              ;BR IF NOT
000456  004737  000000G          CALL    CHKABT          ;SEE IF WE HAVE BEEN ABORTED
000462  016100  000000G          MOV     LSTATE(R1),R0   ;GET JOB'S CURRENT EXECUTION STATE
000466  004737  000000G          CALL    QNSPNX          ;REQUEUE JOB AND CALL JOB SCHEDULER
000472  004737  000000G          CALL    CHKABT          ;SEE IF WE WERE ABORTED WHILE ASLEEP
000476  005302                  4$:     DEC     R2              ;TIME TO CHECK SCHEDULER?
000500  003761                  BLE     17$             ;BR IF YES
000502  004737  000000G          CALL    GETUCH          ;GET CHAR FROM USER'S BUFFER
000506  005700                  TST     R0              ;IS CHAR A NULL?
000510  001402                  BEQ     1$              ;SKIP NULLS
000512  004737  002662'          CALL    PUTCHR          ;PLACE IN USER'S BUFFER
000516  077411                  1$:     SOB     R4,4$           ;BR IF MORE CHARS TO DO
000520  000137  001004'          2$:     JMP     TTFIN
                                ;
                                ;  Use faster routine if buffer is on even byte boundary
                                ;
000524  013704  000006G          9$:     MOV     EMTBLK+6,R4     ;GET NUMBER OF WORDS TO WRITE
000530  012702  000000G          7$:     MOV     #TTCSCH,R2      ;RESET CHAR COUNT FOR CONTINUATION
                                ;  See if we need to interrupt tt output processing to do a job
                                ;  scheduler cycle.
000534  105737  000000G                  TSTB    DOSCHD          ;IS JOB SCHEDULER CYCLE NEEDED?
000540  001410                  BEQ     8$              ;BR IF NOT
000542  004737  000000G          CALL    CHKABT          ;SEE IF WE HAVE BEEN ABORTED
000546  016100  000000G          MOV     LSTATE(R1),R0   ;GET JOB'S CURRENT EXECUTION STATE
```

```
     58 000552  004737  0000000                CALL    QNSPNX          ;REQUEUE JOB AND CALL JOB SCHEDULER
     59 000556  004737  0000000                CALL    CHKABT          ;SEE IF WE WERE ABORTED WHILE ASLEEP
     60 000562  005302                  8$:     DEC     R2              ;TIME TO CHECK FOR SCHEDULER CYCLE?
     61 000564  003761                          BLE     7$              ;BR IF YES
     62 000566  106523                          MFPD    (R3)+           ;GET DATA WORD FROM USER'S BUFFER
     63 000570  111600                          MOVB    (SP),R0         ;GET LOW-ORDER BYTE OF WORD
     64 000572  001402                          BEQ     5$              ;IGNORE IT IF IT IS NULL
     65 000574  004737  002662'                 CALL    PUTCHR          ;SEND CHAR TO TERMINAL
     66 000600  012600                  5$:     MOV     (SP)+,R0        ;GET DATA WORD
     67 000602  105000                          CLRB    R0              ;CLEAR LOW-ORDER BYTE
     68 000604  000300                          SWAB    R0              ;GET HIGH-ORDER BYTE TO LOW-ORDER
     69 000606  001402                          BEQ     6$              ;BR IF LOW-ORDER BYTE IS NULL
     70 000610  004737  002662'                 CALL    PUTCHR          ;SEND CHAR TO TERMINAL
     71 000614  077416                  6$:     SOB     R4,8$           ;LOOP IF MORE WORDS TO WRITE
     72 000616  000137  001004'                 JMP     TTFIN           ;FINISHED
```

```
                                        .SBTTL  TTREAD -- .READ emt with input from terminal
                                ;-------------------------------------------------------
                                ;  TTREAD is executed when a .READ was issued directing input
                                ;  from the terminal.
                                ;
000622  013703  0000040        TTREAD: MOV     EMTBLK+4,R3     ;GET BUFFER ADDRESS
000626  010300                  MOV     R3,R0           ;VALIDATE ADDRESS OF BUFFER
000630  004737  0000000'        CALL    VALADB
000634  013704  0000060        MOV     EMTBLK+6,R4     ;GET # WORDS TO READ
000640  006304                  ASL     R4              ;CONVERT TO # BYTES
000642  060400                  ADD     R4,R0           ;GET ADDRESS OF END OF BUFFER
000644  005300                  DEC     R0
000646  004737  0000000'        CALL    VALADB          ;VALIDATE IT
000652  032777  0000000' 000000' BIT     #CS$EOF,@CHNADR ;DID LAST READ GET END OF FILE?
000660  001407                  BEQ     2$              ;BRANCH IF NOT
000662  042777  0000000' 000000' BIC     #CS$EOF,@CHNADR ;RESET END OF FILE FLAG
000670  052737  0000000' 000000' BIS     #CFLAG,EMTPS    ;SET C-BIT IN USER'S PS
000676  000425                  BR      TTZERO
000700  005737  0000020'        2$:     TST     EMTBLK+2        ;READ BLOCK 0?
000704  001004                  BNE     1$              ;BRANCH IF NOT BLOCK 0
000706  112700  000136          MOVB    #'^,R0          ;PRINT PROMPT CHARACTER
000712  004737  002662'         CALL    PUTCHR
000716  004737  005756'         1$:     CALL    GETCHR          ;GET NEXT INPUT CHAR
000722  120027  0000000'        CMPB    R0,#CTRLZ       ;CTRL-Z ==> END OF FILE
000726  001406                  BEQ     TTEOF           ;BRANCH IF END OF FILE
000730  105700                  4$:     TSTB    R0              ;IGNORE NULLS
000732  001771                  BEQ     1$
000734  004737  0000000'        CALL    PUTUCH          ;MOVE CHAR TO USER'S BUFFER
000740  077412                  SOB     R4,1$           ;BR IF MORE ROOM LEFT
                                ;  NORMAL END OF READ
000742  000420                  BR      TTFIN
                                ;  END OF FILE ON INPUT FROM TT:
000744  052777  0000000' 000000' TTEOF:  BIS     #CS$EOF,@CHNADR ;REMEMBER END OF FILE HIT
                                ;
                                ;  Hit end of file -- Fill remainder of buffer with nulls
                                ;
000752  005704                  TTZERO: TST     R4              ;ROOM LEFT IN BUFFER?
000754  001413                  BEQ     TTFIN           ;BR IF NOT
000756  010300                  1$:     MOV     R3,R0           ;GET NEXT CHAR POSITION
000760  163700  0000040        SUB     EMTBLK+4,R0     ;GET # CHARACTERS TRANSFERRED SO FAR
000764  001403                  BEQ     2$              ;BR IF HAVEN'T GOTTEN ANY CHARS AT ALL
000766  032700  000777          BIT     #777,R0         ;FILLED TO BLOCK BOUNDARY?
000772  001404                  BEQ     TTFIN           ;BR IF YES
000774  005000                  2$:     CLR     R0              ;STORE NULL TO FILL OUR BLOCK
000776  004737  0000000'        CALL    PUTUCH          ;MOVE NULL TO USER'S BUFFER
001002  077413                  SOB     R4,1$           ;LOOP TO FILL REST OF USER'S BUFFER
                                ;  Fall into TTFIN
```

```
                                        ;
                                        ;  FINISHED WITH TT READ/WRITE
                                        ;
001004  010304                  TTFIN:  MOV     R3,R4           ;GET CURRENT BUFFER POINTER
001006  163704  0000040                 SUB     EMTBLK+4,R4     ;GET # BYTES TRANSFERED
001012  006204                          ASR     R4              ;CONVERT TO # WORDS
001014  103001                          BCC     2$              ;BR IF NO ODD BYTE
001016  005204                          INC     R4
001020  010437  0000000         2$:     MOV     R4,URO          ;RETURN # WORDS IN USER'S R0
                                        ;
                                        ;  See if we need to call user's completion routine.
                                        ;
001024  023727  0000100 000001          CMP     EMTBLK+10,#1    ;COMPLETION ROUTINE SPECIFIED?
001032  101415                          BLOS    1$              ;BR IF NOT
                                        ;  Enter a request to call user's completion routine.
001034  004737  0000000                 CALL    FAKCMP          ;QUEUE A COMPLETION REQUEST
001040  105737  0000000                 TSTB    DOSCHD          ;IS JOB SCHEDULER CYCLE NEEDED?
001044  001410                          BEQ     1$              ;BR IF NOT
001046  004737  0000000                 CALL    CHKABT          ;SEE IF WE HAVE BEEN ABORTED
001052  016100  0000000                 MOV     LSTATE(R1),R0   ;GET JOB'S CURRENT EXECUTION STATE
001056  004737  0000000                 CALL    QNSPNX          ;REQUEUE JOB AND CALL JOB SCHEDULER
001062  004737  0000000                 CALL    CHKABT          ;SEE IF WE WERE ABORTED WHILE ASLEEP
                                        ;
                                        ;  Finished
                                        ;
001066  000137  0000000         1$:     JMP     EMTXIT
```

.PRINT

```
                                        .SBTTL  .PRINT
                                ;-----------------------------------------------
                                ;  PROCESS THE .PRINT EMT
                                ;
001072  013702  000000G         PRINT:  MOV     @R0,R2          ;GET ADDRESS OF STRING TO PRINT
001076  020237  000000G                 CMP     R2,@#HIMEM      ;IS BUFFER ADDRESS IN NORMAL JOB REGION?
001102  101403                         BLOS    1$              ;BR IF YES
001104  010200                         MOV     R2,R0           ;VALIDATE ADDRESS OF BUFFER
001106  004737  000000G                 CALL    VALADB
001112  004737  002460'         1$:     CALL    @#OTSTR         ;PRINT THE STRING FROM USER'S BUFFER
001116  000137  000000G                 JMP     EMTXIT
```

```
                        .SBTTL  .TTYOUT
                ;-----------------------------------------------------
                ;  Process the .TTYOUT EMT
                ;
001122          TTYOUT:
                ;
                ;  See if output ring buffer is full
                ;
001122  026127  000000G 000010          CMP     LOTSPC(R1),#8.  ;Is there plenty of free space in output buf?
001130  101022                          BHI     3$              ;Br if yes
                ;
                ;  Output buffer is full.
                ;  See if user wants to suspend job or have carry flag set on return
                ;
001132  032761  000000G 000000G         BIT     #NOWAIT,LJSW(R1);Did he request nowait TT I/O?
001140  001416                          BEQ     3$              ;Br if not
001142  032761  000000G 000000G         BIT     #$NOWTT,LSW5(R1);Did he enable no-wait TT I/O?
001150  001412                          BEQ     3$              ;Br if not
                ;
                ;  Output buffer is full and user has enable no-wait mode.
                ;  See if instruction following EMT 341 is a BCS .-2
                ;
001152  013700  000000G                 MOV     EMTADR,R0       ;Get address of EMT 341 instruction
001156  006560  000002                  MFPI    2(R0)           ;Fetch following instruction
001162  022627  103776                  CMP     (SP)+,#103776   ;BCS .-2 instruction?
001166  001403                          BEQ     3$              ;Br if yes -- Don't return if he will spin
                ;
                ;  Return with carry flag set, and error code 0 to signal that output
                ;  buffer is full.
                ;
001170  005000                          CLR     R0              ;Return error code 0
001172  000137  000000G                 JMP     SETERR
                ;
                ;  Transmit the character (wait if output buffer is full)
                ;
001176  013700  000000G         3$:     MOV     UR0,R0          ;GET THE CHAR TO SEND
001202  032761  000000G 000000G         BIT     #$HITTY,LSW4(R1);ARE WE IN HIGH EFFICIENCY MODE?
001210  001003                          BNE     1$              ;BR IF YES
001212  004737  002662'                  CALL    PUTCHR          ;SEND THE CHAR
001216  000402                          BR      2$
001220  004737  003672'         1$:     CALL    HIPUT           ;SEND CHAR IN HIGH EFFICIENCY MODE
001224  000137  000000G         2$:     JMP     EMTXIT
```

```
                                        .SBTTL  .TTYIN
                                ;-------------------------------------------------------
                                ;  Process the .TTINR EMT
                                ;
001230  032761  000000@ 000000@ TTYIN:  BIT     #NOWAIT,LJSW(R1);DID HE REQUEST NOWAIT .TTYIN?
001236  001445                        BEQ     1$              ;BR IF NOT
                                ;
                                ;  User set JSW flag which says he wants the c-flag set and
                                ;  immediate return if there are no characters available.
                                ;  We will do this once only per end of input.
                                ;  See if any activation chars are pending.
                                ;
001240  005761  000000@                 TST     LACTIV(R1)      ;ANY ACTIVATION CHARS PENDING?
001244  001042                         BNE     1$              ;BR IF YES
001246  004737  007760'                CALL    CFTEST          ;IS INPUT COMING FROM A COMMAND FILE?
001252  103037                         BCC     1$              ;BRANCH IF YES
001254  032761  000000@ 000000@        BIT     #$LOFCF,LSW9(R1);Are we processing a logoff command file?
001262  001405                         BEQ     4$              ;Br if not
001264  052761  000000@ 000000@        BIS     #$DOOFF,LSW(R1) ;Force job logoff
001272  004737  000000@                CALL    STOP            ;Stop job execution
001276  042761  000000@ 000000@ 4$:    BIC     #$NOIN,LSW3(R1) ;ALLOW TERMINAL INPUT TO OCCUR
001304  032761  000000@ 000000@        BIT     #$DBGMD,LSW6(R1);IS DEBUGGER DOING TERMINAL INPUT?
001312  001004                         BNE     3$              ;BR IF YES -- WAIT FOR ACTIVATION CHAR
001314  032761  000000@ 000000@        BIT     #$NOWTT,LSW5(R1);DID HE ENABLE NO-WAIT TT INPUT?
001322  001007                         BNE     2$              ;IF YES THEN SIGNAL NO CHARS AVAILABLE
001324  032761  000000@ 000000@ 3$:    BIT     #$FLAGC,LSW4(R1);HAVE WE ALREADY TOLD HIM ONCE?
001332  001007                         BNE     1$              ;BR IF YES (WAIT FOR INPUT NOW)
001334  052761  000000@ 000000@        BIS     #$FLAGC,LSW4(R1);REMEMBER THAT WE HAVE TOLD HIM
001342  004737  011346'        2$:     CALL    DFRREL          ;RELEASE DEFERRED ECHO MODE
001346  000137  000000@                JMP     SETC            ;RETURN WITH C-FLAG SET
                                ;
                                ;  Get a character
                                ;
001352  042761  000000@ 000000@ 1$:     BIC     #$FLAGC,LSW4(R1);HAVEN'T TOLD HIM NO CHARS AVAILABLE
001360  004737  005756'                CALL    GETCHR          ;GO GET A CHARACTER
001364  010037  000000@                MOV     R0,UR0          ;MOVE TO USER'S R0
001370  000137  000000@                JMP     EMTXIT
```

```
                                        .SBTTL  High-efficiency TTY EMTs
                                ;-----------------------------------------------------------------
                                ;  TURN HIGH-EFFICIENCY TTY MODE ON OR OFF.
                                ;
001374  105737  000000G        XHISET: TSTB    EMTBLK          ;TURN MODE ON OR OFF?
001400  001403                        BEQ     1$              ;BR TO TURN IF OFF
001402  004737  005374'                CALL    HION            ;TURN ON HIGH-EFFICIENCY MODE
001406  000403                        BR      HIRTN
001410  042761  000000G 000000G 1$:    BIC     #$HITTY,LSW4(R1);TURN HIGH-EFFICIENCY MODE OFF
001416  000137  000000G        HIRTN:  JMP     EMTXIT
                                ;
                                ;-----------------------------------------------------------------
                                ;  HIGH-EFFICIENCY OUTPUT.
                                ;
001422  013703  000002G        XHIOUT: MOV     EMTBLK+2,R3     ;BUFFER ADDRESS
001426  010300                        MOV     R3,R0           ;VALIDATE THE ADDRESS
001430  004737  000000G                CALL    VALADB
001434  013704  000004G                MOV     EMTBLK+4,R4     ;# BYTES TO SEND
001440  001766                        BEQ     HIRTN           ;BR IF NO CHARACTERS TO SEND
001442  032761  000000G 000000G        BIT     #$HITTY,LSW4(R1);ARE WE SENDING IN HIGH EFFICIENCY MODE?
001450  001406                        BEQ     1$              ;BR IF NOT
001452  004737  000000G        2$:     CALL    GETUCH          ;GET CHAR FROM USER'S BUFFER
001456  004737  003672'                CALL    HIPUT           ;SEND CHAR IN HIGH EFFICIENCY MODE
001462  077405                        SOB     R4,2$
001464  000754                        BR      HIRTN
001466  004737  000000G        1$:     CALL    GETUCH          ;GET CHAR FROM USER'S BUFFER
001472  004737  002662'                CALL    PUTCHR          ;SEND THE CHARACTER
001476  077405                        SOB     R4,1$
001500  000746                        BR      HIRTN
                                ;
                                ;-----------------------------------------------------------------
                                ;  HIGH-EFFICIENCY TTY INPUT
                                ;
001502  013703  000002G        XHIIN:  MOV     EMTBLK+2,R3     ;ADDRESS OF USER'S BUFFER
001506  010300                        MOV     R3,R0           ;VALIDATE THE ADDRESS
001510  004737  000000G                CALL    VALADB
001514  013704  000004G                MOV     EMTBLK+4,R4     ;SIZE OF BUFFER
001520  060400                        ADD     R4,R0           ;GET ADDRESS OF END OF BUFFER
001522  005300                        DEC     R0
001524  004737  000000G                CALL    VALADB          ;VALIDATE THE ADDRESS
001530  005037  000000G                CLR     UR0             ;RETURN # CHARS GOTTEN IN R0
001534  004737  005756'        4$:     CALL    GETCHR          ;GET NEXT CHARACTER
001540  004737  000000G                CALL    PUTUCH          ;MOVE CHAR TO USER'S BUFFER
001544  005237  000000G                INC     UR0             ;COUNT CHARACTERS IN USER'S R0
001550  005761  000000G                TST     LACTIV(R1)      ;ARE THERE ANY PENDING ACTIVATION CHARS?
001554  001720                        BEQ     HIRTN           ;WE ARE DONE IF NOT
                                ;  NOT ACTIVATION CHARACTER.  SEE IF BUFFER IS FULL.
001556  077412                1$:     SOB     R4,4$           ;LOOP IF ROOM LEFT IN BUFFER
                                ;  OVERFLOWED USER'S BUFFER.  SET C-FLAG AND RETURN.
001560  000417                        BR      XTCC            ;RETURN WITH C-FLAG SET
                                ;
                                ;-----------------------------------------------------------------
                                ;  CHECK FOR TT INPUT ERRORS.
                                ;
001562  105737  000000G        XTERCK: TSTB    EMTBLK          ;CHECK FOR ERRORS?
001566  001405                        BEQ     1$              ;BR IF YES
001570  016137  000000G 000000G        MOV     LINCNT(R1),UR0  ;PUT # OF INPUT CHARS PENDING IN USER'S R0
```

```
     58 001576  000137  000000G                    JMP     EMTXIT
     59 001602  032761  000000G 000000G 1$:        BIT     #$TTERR,LSW4(R1);DID ANY TT INPUT ERRORS OCCUR?
     60 001610  001702                             BEQ     HIRTN           ;BR IF NOT
     61 001612  042761  000000G 000000G            BIC     #$TTERR,LSW4(R1);RESET ERROR FLAG
     62 001620  000137  000000G          XTCC:     JMP     SETC            ;RETURN WITH C-FLAG SET
     63
     64                                 ;---------------------------------------------------------------
     65                                 ;  START TIMER FOR TT READ REQUEST
     66                                 ;
     67 001624  013761  000002G 000000G XRDTIM:   MOV     EMTBLK+2,LRDTIM(R1);SET TT READ TIMEOUT VALUE (0.5 SEC UNITS)
     68 001632  013761  000004G 000000G           MOV     EMTBLK+4,LRTCHR(R1);SET TT TIMEOUT ACTIVATION CHARACTER
     69 001640  052761  100000  000000G           BIS     #100000,LRTCHR(R1) ;SET FLAG SAYING WE HAVE A TIME-OUT CHAR
     70 001646  000137  000000G                    JMP     EMTXIT
```

```
                                        .SBTTL  ASKLIN -- Accept line from terminal
                                ;-----------------------------------------------------------------------
                                ;  ASKLIN is an internal subroutine called from .csispc, .csigen & .gtlin
                                ;  to print a prompt and accept a line of input from the tty or a
                                ;  command file.
                                ;
                                ;  Inputs:
                                ;   R2 = Address of prompt string (in user's area).
                                ;   R3 = Address of buffer where accepted string is to be stored.
                                ;        (Must be in kernel space)
                                ;
001652  010146                  ASKLIN: MOV     R1,-(SP)
001654  010246                          MOV     R2,-(SP)
001656  010346                          MOV     R3,-(SP)
001660  010446                          MOV     R4,-(SP)
001662  010546                          MOV     R5,-(SP)
                                ;
                                ;  Set up buffer pointer and buffer length info
                                ;
001664  010304                          MOV     R3,R4           ;REMEMBER ADDRESS OF START OF BUFFER
001666  010305                          MOV     R3,R5           ;GET ADDRESS OF END OF RECEIVING BUFFER
001670  062705  000120                  ADD     #80.,R5         ;(GETLIN RESTRICTS BUFFER TO 81 CHARS)
001674  113701  000000G                 MOVB    CORUSR,R1       ;GET CURRENT USER INDEX #
001700  052761  000000G 000000G         BIS     #$GTLIN,LSW4(R1);REMEMBER THAT .GTLIN IS BEING DONE
                                ;
                                ;  Determine if we need to process a deferred control-C character
                                ;  that was previously acquired from an expanded CCL command by a
                                ;  non-terminating .GTLIN
                                ;
001706  032761  000000G 000000G         BIT     #$CFDCC,LSW4(R1);DO WE HAVE A DEFERRED ^C?
001714  001404                          BEQ     10$             ;BR IF NOT
001716  032761  000000G 000000G         BIT     #GTLTTY,LJSW(R1);IS THIS A NON-TERMINATING .GTLIN?
001724  001440                          BEQ     11$             ;BR IF NOT -- REPORT CONTROL-C NOW
                                ;
                                ;  If this is a terminating .GTLIN and we pushed a control-C
                                ;  previously due to a non-terminating .GTLIN, halt the execution
                                ;  of the program without printing prompt.
                                ;
001726  032761  000000G 000000G 10$:     BIT     #$NTGCC,LSW9(R1);Did we push a control-C?
001734  001411                          BEQ     20$             ;Br if not
001736  032761  000000G 000000G         BIT     #GTLTTY,LJSW(R1);Is this a non-terminating .GTLIN?
001744  001060                          BNE     15$             ;Br if non-terminating .GTLIN
001746  042761  000000G 000000G         BIC     #$NTGCC,LSW9(R1);Say ctrl-C not pushed
001754  004737  000000G                 CALL    STOP            ;Stop program execution
                                ;
                                ;  Determine if the input is coming from a command file
                                ;
001760  005037  000000G         20$:    CLR     CFSPND          ;No suspended command file yet
001764  004737  007262'         13$:    CALL    GTCFCH          ;TRY TO GET A CHAR FROM COMMAND FILE
001770  103020                          BCC     8$              ;BR IF GOT A CHAR FROM COMMAND FILE
                                ;
                                ;  Input is not coming from a command file.
                                ;  See if we need to reenter Kmon to get next command from IND or
                                ;  from user command processor.
                                ;
001772  032761  000000G 000000G         BIT     #$INKMN,LSW4(R1);ARE WE IN KMON NOW?
002000  001471                          BEQ     3$              ;BR IF NOT
```

```
     58 002002  032761  0000000 0000000          BIT     #$UKMON,LSW7(R1);IS USER COMMAND PROCESSOR ACTIVE?
     59 002010  001006                          BNE     11$             ;BR IF YES -- GET COMMAND FROM IT
     60 002012  013700  0000000                 MOV     CXTRMN,R0       ;GET ADDR OF SIMULATED RMON DATA FOR JOB
     61 002016  132760  0000000 0000000         BITB    #IN$ACT,R$INST(R0);IS INPUT BEING PROVIDED BY IND?
     62 002024  001457                          BEQ     3$              ;BR IF NOT
     63 002026  004737  0000000          11$:   CALL    STOP            ;REENTER KMON AND GET NEXT COMMAND FROM IND
     64                                     ;
     65                                     ;  We got a character from a control file.
     66                                     ;  See if character is a control-C or control-Z.
     67                                     ;  Note: we treat ctrl-Z the same as ctrl-C if KMON is reading file.
     68                                     ;
     69 002032  120027  0000000          8$:    CMPB    R0,#CTRLZ       ;IS CHAR CTRL-Z?
     70 002036  001006                          BNE     16$             ;BR IF NOT
     71 002040  032761  0000000 0000000         BIT     #$INKMN,LSW4(R1);IS KMON READING FILE?
     72 002046  001402                          BEQ     16$             ;BR IF NOT
     73 002050  112700  0000000                 MOVB    #CTRLC,R0       ;TRANSLATE CTRL-Z TO CTRL-C
     74 002054  120027  0000000          16$:   CMPB    R0,#CTRLC       ;IS CHAR CTRL-C?
     75 002060  001033                          BNE     7$              ;BR IF NOT ^C
     76                                     ;
     77                                     ;  Character is control-C.
     78                                     ;
     79                                     ;  There are 3 cases to deal with:
     80                                     ;  1. This is not a non-terminating .GTLIN.  In this case we simply
     81                                     ;     accept the ctrl-C which terminates the execution of the program.
     82                                     ;  2. This is a non-terminating .GTLIN and we are getting characters
     83                                     ;     from an expanded CCL command.  In this case, we defer the ctrl-C
     84                                     ;     processing until the next non-terminating .GTLIN is done.
     85                                     ;  3. This is a non-terminating .GTLIN and we are not getting characters
     86                                     ;     from an expanded CCL command.  In this case we defer the ctrl-C
     87                                     ;     and get characters from the terminal.
     88                                     ;
     89 002062  032761  0000000 0000000          BIT     #GTLTTY,LJSW(R1);IS THIS A NON-TERMINATING .GTLIN?
     90 002070  001427                          BEQ     7$              ;BR IF NOT -- GO TERMINATE PROGRAM
     91 002072  004737  007262'         14$:    CALL    GTCFCH          ;SKIP TO END OF LINE THAT HAS ^C
     92 002076  103403                          BCS     15$
     93 002100  120027  0000000                 CMPB    R0,#LF          ;REACHED END OF LINE?
     94 002104  001372                          BNE     14$             ;LOOP IF NOT
     95 002106  032761  0000000 0000000  15$:    BIT     #$CFCCL,LSW4(R1);IS THIS THE END OF AN EXPANDED CCL COMMAND?
     96 002114  001404                          BEQ     12$             ;BR IF NOT
     97 002116  052761  0000000 0000000         BIS     #$CFDCC,LSW4(R1);REMEMBER WE HAVE A DEFERRED CONTROL-C
     98 002124  000717                          BR      13$             ;GO BACK AND GET NEXT CHAR FOLLOWING CONTROL-C
     99 002126  052761  0000000 0000000  12$:    BIS     #$NTGCC,LSW9(R1);Push a control-C for next .GTLIN
    100 002134  013737  0000000 0000000         MOV     CFPNT,CFSPND    ;SAVE COMMAND FILE POINTER -- Suspend file
    101 002142  004737  010016'                  CALL    CFSTOP          ;Suspend command file input
    102 002146  000406                          BR      3$
    103 002150  110037  0000000          7$:    MOVB    R0,CFHOLD       ;PUSH COMMAND FILE CHAR
    104 002154  032761  0000000 0000000         BIT     #$QUIET,LSW4(R1);ARE WE LISTING THE COMMAND FILE?
    105 002162  001012                          BNE     1$              ;BR IF NOT -- DON'T LIST PROMPT THEN
    106                                     ;
    107                                     ;  List prompt string
    108                                     ;
    109 002164  005702                  3$:    TST     R2              ;Is there a prompt string to print?
    110 002166  001410                          BEQ     1$              ;Br if not
    111 002170  005737  0000000                 TST     CFPNT           ;Is input coming from a command file?
    112 002174  001003                          BNE     21$             ;Br if yes
    113 002176  042761  0000000 0000000         BIC     #$CTRLO,LSW3(R1);Reset control-O
    114 002204  004737  002460'          21$:   CALL    UOTSTR          ;Print the prompt
```

```
                                        ;
                                        ;  Get the input line
                                        ;
002210  005737  000000G         1$:     TST     CFPNT           ;Input coming from a command file?
002214  001003                          BNE     19$             ;Br if yes
002216  052761  000000G 000000G         BIS     #$CFABT,LSW6(R1);If ctrl-C received, abort command files
002224  004737  005756'        19$:     CALL    GETCHR          ;GET AN INPUT CHAR
002230  042761  000000G 000000G         BIC     #$CFABT,LSW6(R1);Clear command file abort flag
002236  120027  000000G                 CMPB    R0,#LF          ;REACHED END OF LINE?
002242  001424                          BEQ     2$              ;BR IF YES
002244  120027  000041                  CMPB    R0,#'!          ;START OF COMMENT?
002250  001005                          BNE     17$             ;BR IF NOT
002252  004737  005756'        18$:     CALL    GETCHR          ;SKIP OVER REST OF COMMENT
002256  120027  000000G                 CMPB    R0,#CR
002262  001373                          BNE     18$             ;LOOP TILL WE REACH END OF COMMENT
002264  020305                  17$:    CMP     R3,R5           ;ARE PAST THE END OF THE BUFFER?
002266  101350                          BHI     1$              ;BR IF YES (DISCARD THE CHARACTER)
002270  110023                          MOVB    R0,(R3)+        ;MOVE CHAR TO BUFFER
002272  120027  000000G                 CMPB    R0,#CTRLZ       ;IS CHAR CTRL-Z?
002276  001344                          BNE     1$              ;BR IF NOT
002300  032761  000000G 000000G         BIT     #$INKMN,LSW4(R1);IS KMON DOING .GTLINE?
002306  001740                          BEQ     1$              ;BR IF NOT KMON
002310  004737  000000G                 CALL    STOP            ;TERMINATE PROGRAM IF CTRL-Z HIT
002314  020304                  2$:     CMP     R3,R4           ;IGNORE THE LF IF IT IS AT START OF LINE
002316  001734                          BEQ     1$
002320  105043                          CLRB    -(R3)           ;REPLACE CR WITH NULL
                                        ;
                                        ;  Finished getting input line
                                        ;
002322  013700  000000G                 MOV     CFSPND,R0       ;GET @FILE POINTER
002326  001404                          BEQ     5$              ;BR IF NO NEED TO REPLACE
002330  004737  010042'                 CALL    CFSTRT          ;Restart command file input
002334  005037  000000G                 CLR     CFSPND          ;SAY THERE IS NO SUSPENDED COMMAND FILE
002340  042761  000000G 000000G 5$:     BIC     #$GTLIN,LSW4(R1);SAY .GTLIN EMT IS FINISHED
002346  012605                          MOV     (SP)+,R5
002350  012604                          MOV     (SP)+,R4
002352  012603                          MOV     (SP)+,R3
002354  012602                          MOV     (SP)+,R2
002356  012601                          MOV     (SP)+,R1
002360  000207                          RETURN
```

```
                                        .SBTTL  OUTSTR -- Print a system message on the user's console
                                ;-----------------------------------------------
                                ;  OUTSTR is called to print a system message on the user's terminal.
                                ;  Inputs:
                                ;   R2 = Address of ASCIZ string to be printed (in kernel space).
                                ;
002362  010046                  OUTSTR: MOV     R0,-(SP)
002364  010146                          MOV     R1,-(SP)
002366  010246                          MOV     R2,-(SP)
002370  113701  0000000G                MOVB    CORUSR,R1       ;GET USER'S INDEX #
002374  112200                  1$:     MOVB    (R2)+,R0        ;GET NEXT CHAR OF MESSAGE
002376  001406                          BEQ     2$              ;BRANCH WHEN END HIT
002400  120027  000200                   CMPB    R0,#200         ;STOP WITHOUT CR-LF?
002404  001413                          BEQ     9$              ;BRANCH IF YES
002406  004737  002662'                 CALL    PUTCHR          ;OUTPUT THE CHARACTER
002412  000770                          BR      1$
002414  112700  000015          2$:     MOVB    #15,R0          ;PUT OUT CR-LF
002420  004737  002662'                 CALL    PUTCHR
002424  112700  000012                  MOVB    #12,R0
002430  004737  002662'                 CALL    PUTCHR
002434  012602                  9$:     MOV     (SP)+,R2
002436  012601                          MOV     (SP)+,R1
002440  012600                          MOV     (SP)+,R0
002442  000207                          RETURN

                                        .SBTTL  CSIMSG -- Print a CSI asciz error message
                                ;-----------------------------------------------------------------------
                                ;  CSIMSG is called to print a CSI asciz error message on the user's
                                ;  terminal.
                                ;
                                ;  Inputs:
                                ;   R2 = CSI error message code.
                                ;
002444                          CSIMSG:
002444  006302                          ASL     R2              ;MULTIPLY BY TWO
002446  016202  000004'                  MOV     CSIERR(R2),R2   ;INDEX TO THE CORRECT ERROR STRING
002452  004737  002362'                 CALL    OUTSTR          ;PRINT THE ERROR MESSAGE
002456  000207                          RETURN                  ;RETURN TO CONTINUE PROCESSING
```

```
                                        .SBTTL  UOTSTR -- print an asciz user's message on the user's console
                                ;-----------------------------------------------------------------------
                                ;  UOTSTR is called to print an asciz message that is stored in the
                                ;  user's area.
                                ;
                                ;  Inputs:
                                ;   R2 = Address of asciz string to be printed (in user's space).
                                ;
002460  010046                  UOTSTR: MOV     R0,-(SP)
002462  010146                          MOV     R1,-(SP)
002464  010246                          MOV     R2,-(SP)
002466  010346                          MOV     R3,-(SP)
002470  010446                          MOV     R4,-(SP)
002472  012704  0000000G              MOV     #TTCSCH,R4      ;SET CHARACTER COUNTER FOR SCHEDULING CALL
002476  113701  0000000G              MOVB    CORUSR,R1       ;GET USER'S INDEX #
002502  012703  000200                MOV     #200,R3         ;GET CHAR USED TO STOP STRING WITHOUT CR-LF
002506  032702  000001                BIT     #1,R2           ;IS BUFFER ON EVEN OR ODD BYTE BOUNDARY?
002512  001420                      BEQ     3$              ;BR IF ON EVEN BYTE BOUNDARY
002514  005302                      DEC     R2              ;POINT TO 1ST BYTE OF WORD
002516  106522                      MFPD    (R2)+           ;GET WORD CONTAINING BYTE
002520  000426                      BR      4$
                                ;
                                ;  See if we need to interrupt output processing for a job scheduling cycle.
                                ;
002522  012704  0000000G          1$:     MOV     #TTCSCH,R4      ;RESET THE CHARACTER COUNT FOR SCHEDULING CALL
002526  105737  0000000G                  TSTB    DOSCHD          ;IS JOB SCHEDULER CYCLE NEEDED?
002532  001410                      BEQ     3$              ;BR IF NOT
002534  004737  0000000G                  CALL    CHKABT          ;SEE IF WE HAVE BEEN ABORTED
002540  016100  0000000G                  MOV     LSTATE(R1),R0   ;GET JOB'S CURRENT EXECUTION STATE
002544  004737  0000000G                  CALL    QNSPNX          ;REQUEUE JOB AND CALL JOB SCHEDULER
002550  004737  0000000G                  CALL    CHKABT          ;SEE IF WE WERE ABORTED WHILE ASLEEP
002554  005304                  3$:     DEC     R4              ;TIME TO CHECK FOR SCHEDULER CYCLE?
002556  003761                      BLE     1$              ;BR IF YES
002560  106522                      MFPD    (R2)+           ;GET WORD WITH NEXT 2 BYTES OF STRING
002562  111600                      MOVB    (SP),R0         ;GET NEXT BYTE OF STRING
002564  001417                      BEQ     5$              ;BR IF NULL AT END OF STRING HIT
002566  120003                      CMPB    R0,R3           ;IS CHAR #200?
002570  001413                      BEQ     10$             ;IF YES THEN END STRING WITHOUT CRLF
002572  004737  002662'              CALL    PUTCHR          ;SEND CHAR TO TERMINAL
002576  012600                  4$:     MOV     (SP)+,R0        ;GET WORD WITH CHAR IN HIGH-ORDER BYTE
002600  105000                      CLRB    R0              ;CLEAR LOW-ORDER BYTE
002602  000300                      SWAB    R0              ;MOVE HIGH-ORDER BYTE TO LOW-ORDER
002604  001410                      BEQ     2$              ;BR IF NULL AT END OF STRING HIT
002606  120003                      CMPB    R0,R3           ;IS THIS 200 AT END OF STRING?
002610  001416                      BEQ     9$              ;BR IF YES
002612  004737  002662'              CALL    PUTCHR          ;SEND CHAR TO TERMINAL
002616  000756                      BR      3$              ;GO SEND MORE CHARS
002620  005726                  10$:    TST     (SP)+           ;CLEAN OFF STACK
002622  000411                      BR      9$
002624  005726                  5$:     TST     (SP)+           ;CLEAN OFF STACK
002626  112700  000015          2$:     MOVB    #15,R0          ;PUT OUT CR-LF
002632  004737  002662'              CALL    PUTCHR
002636  112700  000012              MOVB    #12,R0
002642  004737  002662'              CALL    PUTCHR
002646  012604                  9$:     MOV     (SP)+,R4
002650  012603                          MOV     (SP)+,R3
002652  012602                          MOV     (SP)+,R2
```

```
     58 002654  012601                          MOV     (SP)+,R1
     59 002656  012600                          MOV     (SP)+,R0
     60 002660  000207                          RETURN
```

```
                                        .SBTTL
                                        .SBTTL  **  Program Level Output Character Processing  **
                                        .SBTTL  PUTCHR -- Send character to terminal
                                ;-----------------------------------------------------------------------
                                ;  PUTCHR is called to queue a character to be sent to a terminal.
                                ;
                                ;  Inputs:
                                ;   R0 = Character to be sent.
                                ;   R1 = Virtual line number.
                                ;
002662                          PUTCHR:
                                ;
                                ;  If line is in "high efficiency" mode, bypass a lot of normal
                                ;  processing and use the high efficiency version of PUTCHR.
                                ;
002662  032761  0000000 0000000         BIT     #$HITTY,LSW4(R1);Is this line in high efficiency mode?
002670  001402                          BEQ     1$              ;Br if not
002672  000137  003672'                 JMP     HIPUT           ;Use high efficiency version of PUTCHR
                                ;
                                ;  Return immediately if the line has disconnected
                                ;
002676  032761  000000C 0000000 1$:     BIT     #<$DISCN!$CTRLC>,LSW(R1) ;Has job disconnected?
002704  001406                          BEQ     2$              ;Br if not
002706  032761  0000000 0000000         BIT     #$NOIN,LSW3(R1) ;Are we doing logoff processing now?
002714  001057                          BNE     9$              ;Br if yes -- return immediately from PUTCHR
002716  004737  0000000               CALL    STOP            ;Terminate execution of job and enter KMON
                                ;
                                ;  Mask character to 7 or 8 bits depending on setting of EIGHTBIT option.
                                ;
002722  042700  177400          2$:    BIC     #^C<377>,R0     ;Mask character to 8 bits
002726  032761  0000000 0000000         BIT     #$8BIT,LSW2(R1) ;Is eightbit support option selected?
002734  001002                          BNE     8$              ;Br if yes
002736  042700  177600                  BIC     #^C<177>,R0     ;Mask character to 7 bits
                                ;
                                ;  See if we should suppress output from programs such as
                                ;  EDIT, TECO, and DIBOL which try to echo characters themselves.
                                ;
002742  032761  0000000 0000000 8$:     BIT     #$NOOUT,LSW3(R1);Is output echo suppression in effect?
002750  001403                          BEQ     3$              ;Br if not
002752  004737  004052'                 CALL    ESCHK           ;See if we should discard this character
002756  103436                          BCS     9$              ;Br if we should discard this character
                                ;
                                ;  Remember the last character output to this line
                                ;
002760  110061  0000000         3$:     MOVB    R0,LSNDCH(R1)   ;Remember last character sent to line
                                ;
                                ;  If character is being sent in transparent mode, set transparent
                                ;  flag for this character
                                ;
002764  032761  0000000 0000000         BIT     #$TRNSP,LSW3(R1);Is line is transparency mode?
002772  001403                          BEQ     4$              ;Br if not
002774  052700  0000000               BIS     #TRNSFL,R0      ;Set transparency flag for this character
003000  000415                          BR      5$
                                ;
                                ;  Determine if this character is a leadin character or part of
                                ;  a lead-in function sequence.
                                ;
```

```
     58 003002  005761  000000G          4$:     TST     LTSCMD(R1)          ;Are we processing a lead-in function now?
     59 003006  001403                           BEQ     6$                  ;Br if not
     60 003010  004737  004250'                  CALL    LIFUN               ;Process the lead-in function
     61 003014  000417                           BR      9$                  ;Finished with character
     62 003016  120037  000000G          6$:     CMPB    R0,VTSLCH           ;Is this the lead-in character?
     63 003022  001004                           BNE     5$                  ;Br if not
     64 003024  012761  004304' 000000G         MOV     #GTCM1,LTSCMD(R1);Say we are starting a lead-in function
     65 003032  000410                           BR      9$                  ;Finished with character
     66                                ;
     67                                ;  See if we should write this character to the terminal log file
     68                                ;
     69 003034  032737  000000G 000000G 5$:     BIT     #LF$OUT,LOGFLG      ;Are we logging output characters?
     70 003042  001402                           BEQ     7$                  ;Br if not
     71 003044  004737  010714'                  CALL    LOGCHR              ;Log this character
     72                                ;
     73                                ;  We have done the highest level processing associated with
     74                                ;  PUTCHR.  Now call PUTCH1 to do the next level of processing.
     75                                ;
     76 003050  004737  003056'          7$:     CALL    PUTCH1              ;Queue character for terminal
     77                                ;
     78                                ;  Finished
     79                                ;
     80 003054  000207                   9$:     RETURN
```

```
                                        .SBTTL  PUTCH1 -- Queue character for terminal
                                ;-----------------------------------------------------------------------
                                ;  PUTCH1 is the second level of character queueing routines.
                                ;  Calling this routine rather than PUTCHR bypasses the following
                                ;  character processing operations:
                                ;
                                ;   1. Stop the job if it is disconnected.
                                ;   2. Echo suppression.
                                ;   3. Transparent character flagging.
                                ;   4. TSX lead-in function processing.
                                ;   5. Terminal logging.
                                ;
                                ;  Inputs:
                                ;   R0 = Character to be queued for the terminal.
                                ;   R1 = Virtual line number of the terminal.
                                ;
003056                          PUTCH1:
                                ;
                                ;  If this is a detached job, discard its output
                                ;
003056  032761  0000000 0000000         BIT     #$DETCH,LSW(R1) ;Is this a detached job?
003064  001015                          BNE     9$              ;Br if yes -- discard its output
                                ;
                                ;  If we are in a command file and program output is being suppressed,
                                ;  discard this character.
                                ;
003066  032761  0000000 0000000         BIT     #$CFSOT,LSW4(R1);Should we suppress command file output?
003074  001403                          BEQ     1$              ;Br if not
003076  005737  0000000               TST     CFPNT           ;Is input coming from a command file?
003102  001006                          BNE     9$              ;Br if yes -- discard this character
                                ;
                                ;  If Control-O has been typed, discard this character
                                ;
003104  032761  0000000 0000000 1$:    BIT     #$CTRLO,LSW3(R1);Has ctrl-O been typed?
003112  001002                          BNE     9$              ;Br if yes -- discard this character
                                ;
                                ;  We want to queue this character for the terminal.
                                ;  Call PUTCH2 to do the next level of processing
                                ;
003114  004737  003122'                 CALL    PUTCH2          ;Queue character for the terminal
                                ;
                                ;  Finished
                                ;
003120  000207                  9$:     RETURN
```

```
                                        .SBTTL  PUTCH2 -- Queue character for a terminal
                                ;-----------------------------------------------------------------------
                                ;  PUTCH2 is the third level of character processing associated with
                                ;  PUTCHR.  Calling PUTCH2 rather than PUTCH1 bypasses the following
                                ;  character processing operations:
                                ;
                                ;   1.  Discarding output for detached jobs.
                                ;   2.  Suppressing output while inside command file.
                                ;   3.  Suppressing output if Control-O typed.
                                ;
                                ;  Inputs:
                                ;   R0 = Character to be queued for the terminal.
                                ;   R1 = Virtual line index number.
                                ;
003122  010246                  PUTCH2: MOV     R2,-(SP)
                                ;
                                ;  If character is being sent in transparency mode, by pass
                                ;  keeping track of its position.
                                ;
003124  032700  0000000                 BIT     #TRNSFL,R0      ;Is character in transparency mode?
003130  001015                          BNE     2$              ;Br if yes
                                ;
                                ;  Determine if this is a regular or control character.
                                ;
003132  020027  000037          3$:     CMP     R0,#37          ;Is this a control character?
003136  101005                          BHI     1$              ;Br if not
                                ;
                                ;  This is a control character.
                                ;  Call appropriate routine to process it.
                                ;
003140  010002                          MOV     R0,R2           ;Get character
003142  006302                          ASL     R2              ;Convert character to word table index
003144  004772  003174'                 CALL    @PC2VEC(R2)     ;Call processing routine
003150  000407                          BR      9$              ;Finished with character
                                ;
                                ;  This is a regular character.
                                ;  Keep track of cursor position.
                                ;
003152  120027  0000000         1$:     CMPB    R0,#RUBOUT      ;Is this a rubout character?
003156  001402                          BEQ     2$              ;Br if yes
003160  105261  0000000                 INCB    LCOL(R1)        ;Advance cursor column position
                                ;
                                ;  Insert this character into the terminal buffer
                                ;
003164  004737  003462'         2$:     CALL    QUECHR          ;Insert character into terminal buffer
                                ;
                                ;  Finished
                                ;
003170  012602                  9$:     MOV     (SP)+,R2
003172  000207                          RETURN
```

```
                                ;-----------------------------------------------------------------------
                                ;  Vector of addresses of character processing routines for
                                ;  control characters encountered by PUTCH2.
                                ;
                                ;  On entry to the processing routine, the following registers will
                                ;  be set up:
                                ;   R0 = Character
                                ;   R1 = Virtual line number
                                ;
003174  003274'         PC2VEC: .WORD   PCCINS          ; 00 - NUL
003176  003274'                 .WORD   PCCINS          ; 01 - SOH
003200  003274'                 .WORD   PCCINS          ; 02 - STX
003202  003274'                 .WORD   PCCINS          ; 03 - ETX
003204  003274'                 .WORD   PCCINS          ; 04 - EOT
003206  003274'                 .WORD   PCCINS          ; 05 - ENQ
003210  003274'                 .WORD   PCCINS          ; 06 - ACK
003212  003274'                 .WORD   PCCINS          ; 07 - BEL
003214  003302'                 .WORD   PCCBS           ; 10 - BS
003216  003334'                 .WORD   PCCHT           ; 11 - HT
003220  003274'                 .WORD   PCCINS          ; 12 - LF
003222  003274'                 .WORD   PCCINS          ; 13 - VT
003224  003420'                 .WORD   PCCFF           ; 14 - FF
003226  003322'                 .WORD   PCCCR           ; 15 - CR
003230  003274'                 .WORD   PCCINS          ; 16 - SO
003232  003274'                 .WORD   PCCINS          ; 17 - SI
003234  003274'                 .WORD   PCCINS          ; 20 - DLE
003236  003274'                 .WORD   PCCINS          ; 21 - DC1
003240  003274'                 .WORD   PCCINS          ; 22 - DC2
003242  003274'                 .WORD   PCCINS          ; 23 - DC3
003244  003274'                 .WORD   PCCINS          ; 24 - DC4
003246  003274'                 .WORD   PCCINS          ; 25 - NAK
003250  003274'                 .WORD   PCCINS          ; 26 - SYN
003252  003274'                 .WORD   PCCINS          ; 27 - ETB
003254  003274'                 .WORD   PCCINS          ; 30 - CAN
003256  003274'                 .WORD   PCCINS          ; 31 - EM
003260  003274'                 .WORD   PCCINS          ; 32 - SUB
003262  003274'                 .WORD   PCCINS          ; 33 - ESC
003264  003274'                 .WORD   PCCINS          ; 34 - FS
003266  003274'                 .WORD   PCCINS          ; 35 - GS
003270  003274'                 .WORD   PCCINS          ; 36 - RS
003272  003274'                 .WORD   PCCINS          ; 37 - US
```

```
                                ;-----------------------------------------------------------------------
                                ;  Processing routines for control characters sent to terminal through
                                ;  PUTCH2.
                                ;
                                ;  Normal control character.
                                ;  Insert into terminal buffer but do not affect cursor position.
                                ;
003274  004737  003462'         PCCINS: CALL    QUECHR          ;Put char into terminal buffer
003300  000207                          RETURN
                                ;
                                ;  Backspace -- Backup cursor position
                                ;
003302  105361  000000@         PCCBS:  DECB    LCOL(R1)        ;Backup cursor position
003306  002002                          BGE     1$              ;Br if didn't go past front of line
003310  105061  000000@                 CLRB    LCOL(R1)        ;Don't allow to go to left of line start
003314  004737  003462'         1$:     CALL    QUECHR          ;Queue the character
003320  000207                          RETURN
                                ;
                                ;  Carriage return
                                ;
003322  105061  000000@         PCCCR:  CLRB    LCOL(R1)        ;Say cursor is at left margin
003326  004737  003462'                 CALL    QUECHR          ;Queue the character
003332  000207                          RETURN
                                ;
                                ;  Tab -- Translate to spaces if tab simulation wanted
                                ;
003334  010246                  PCCHT:  MOV     R2,-(SP)
003336  116102  000000@                 MOVB    LCOL(R1),R2     ;Get current column position
003342  032761  000000@ 000000@         BIT     #$TAB,LSW2(R1)  ;Should we simulate tabs?
003350  001011                          BNE     1$              ;Br if not
003352  112700  000040                  MOVB    #' ,R0          ;Get space character
003356  004737  003462'         2$:     CALL    QUECHR          ;Send a space
003362  005202                          INC     R2              ;Advance cursor position
003364  032702  000007                  BIT     #7,R2           ;Are we up to next tab stop?
003370  001372                          BNE     2$              ;Loop if not
003372  000406                          BR      9$
003374  004737  003462'         1$:     CALL    QUECHR          ;Send the tab character to the terminal
003400  062702  000010                  ADD     #8.,R2          ;Bound up to next tab stop
003404  042702  000007                  BIC     #7,R2
003410  110261  000000@         9$:     MOVB    R2,LCOL(R1)     ;Save new cursor position
003414  012602                          MOV     (SP)+,R2
003416  000207                          RETURN
                                ;
                                ;  Form feed -- See if we should translate to spaces
                                ;
003420  032761  000000@ 000000@ PCCFF:  BIT     #$FORM,LSW2(R1) ;Should we simulate form-feeds?
003426  001012                          BNE     2$              ;Br if not
003430  010246                          MOV     R2,-(SP)
003432  012702  000010                  MOV     #8.,R2          ;Put out 8 LF characters
003436  112700  000000@                 MOVB    #LF,R0          ;Get line feed character
003442  004737  003462'         1$:     CALL    QUECHR          ;Send a line feed
003446  077203                          SOB     R2,1$           ;Loop till all line feeds sent
003450  012602                          MOV     (SP)+,R2
003452  000402                          BR      9$
003454  004737  003462'         2$:     CALL    QUECHR          ;Send FF character to terminal
003460  000207                  9$:     RETURN
```

```
                                        .SBTTL  QUECHR -- Queue character for transmission
                                ;-----------------------------------------------------------------------
                                ;  QUECHR is called to queue a character for transmission.
                                ;  If the job has display windowing turned on, the character is passed
                                ;  to the window manager before being queued for transmission.
                                ;
                                ;  Inputs:
                                ;   R0 = Character to be queued.
                                ;   R1 = Virtual terminal number.
                                ;
003462                          QUECHR:
                                ;
                                ;  See if display windowing is turned on for this line
                                ;
003462  005761  0000000                 TST     LWINDO(R1)      ;Is job doing display windowing?
003466  001410                          BEQ     1$              ;Br if not
003470  032761  0000000 0000000         BIT     #$NOWIN,LSW11(R1);Are we suppressing windowing now?
003476  001004                          BNE     1$              ;Br if yes
003500                                  OCALL   WINCHR          ;Pass character to window manager
003506  103402                          BCS     9$              ;Br if we should not display this char
                                ;
                                ;  Queue the character for transmission
                                ;
003510  004737  003516'         1$:     CALL    BUFCHR          ;Queue character for transmission
                                ;
                                ;  Finished
                                ;
003514  000207                  9$:     RETURN
```

```
                                        .SBTTL  BUFCHR -- Insert char or suspend if full
                                ;-----------------------------------------------------------------------
                                ;  BUFCHR is called to queue a character for a terminal.
                                ;  If there is adequate space in the terminal output buffer the character
                                ;  is inserted.  If there is not adequate space the job is suspended
                                ;  until space becomes available in the output terminal buffer.
                                ;
                                ;  Inputs:
                                ;   R0 = Character to be queued.
                                ;   R1 = Virtual terminal number.
                                ;
003516  010246                  BUFCHR: MOV     R2,-(SP)
                                ;
                                ;  Disable interrupts
                                ;
003520                          1$:     DISABL                  ;** Disable **
                                ;
                                ;  See if there is room in the output buffer for the character
                                ;
003526  026127  0000000 000010          CMP     LOTSPC(R1),#8.  ;Is there plenty of free space in buffer?
003534  101016                          BHI     2$              ;Br if yes
003536  105737  0000000                 TSTB    FRKPRI          ;Are we running at interrupt level? (echoing)
003542  001404                          BEQ     3$              ;Br if not at interrupt level
003544  005761  0000000                 TST     LOTSPC(R1)      ;Allow char echoing to use all of buffer
003550  003010                          BGT     2$              ;Br if buffer not completely full
003552  000442                          BR      9$              ;Discard char if no space for echo
                                ;
                                ;  Output buffer is full.
                                ;  Suspend job's execution and wait for buffer space to becom available.
                                ;
003554  004737  005674'          3$:     CALL    PCSPND          ;Suspend execution of job ** Enable **
003560  032761  0000000 0000000         BIT     #$CTRLO,LSW3(R1);Was Ctrl-O typed while we were asleep?
003566  001754                          BEQ     1$              ;Br if not
003570  000433                          BR      9$              ;Discard character if ctrl-O typed
                                ;
                                ;  There is room for the character in the output buffer
                                ;  Insert character into the buffer
                                ;
003572  005361  0000000         2$:     DEC     LOTSPC(R1)      ;Decrease free space count for buffer
003576  016102  0000000                 MOV     LOTNXT(R1),R2   ;Get pointer to next free cell in buffer
003602                                  TTMAP                   ;Map kernel PAR 6 to TT buffer
003616  110022                          MOVB    R0,(R2)+        ;Move character into TT buffer
003620                                  UNMAP                   ;Remap kernel PAR 6 to job context block
003626  020261  0000000                 CMP     R2,LOTEND(R1)   ;Did we move beyond end of buffer?
003632  103402                          BLO     4$              ;Br if not
003634  016102  0000000                 MOV     LOTBUF(R1),R2   ;Wrap around to front of buffer
003640  010261  0000000         4$:     MOV     R2,LOTNXT(R1)   ;Save next-character pointer
                                ;
                                ;  Start transmitter for this line
                                ;
003644                                  ENABL                   ;** Enable **
003652  004777  0000000                 CALL    @TRNSTR         ;Start transmitter for this line
003656  000403                          BR      10$
                                ;
                                ;  Finished
                                ;
003660                          9$:     ENABL                   ;** Enable **
```

```
     58 003666  012602                 10$:    MOV     (SP)+,R2
     59 003670  000207                         RETURN
```

```
                                        .SBTTL  HIPUT  -- High efficiency PUTCHR
                                ;-----------------------------------------------------------------------
                                ;  HIPUT -- Routine to move a character to the terminal output buffer
                                ;  and print it in high efficiency mode.
                                ;
                                ;  Inputs:
                                ;   R0 = Character to be sent.
                                ;   R1 = Virtual line number of job.
                                ;
003672                          HIPUT:;
003672  010246                          MOV     R2,-(SP)
                                ;
                                ;  If this is a detached job, discard its output
                                ;
003674  032761  0000000 0000000         BIT     #$DETCH,LSW(R1) ;Is this a detached job?
003702  001061                          BNE     9$              ;Br if yes -- discard its output
003704  032761  0000000 0000000         BIT     #<$DISCN+$CTRLC+$DETCH>,LSW(R1) ;IS LINE DISCONNECTED?
003712  001402                          BEQ     1$              ;BR IF NOT
003714  004737  0000000                 CALL    STOP            ;STOP JOB
                                ;
                                ;  See if display windowing is turned on for this line
                                ;
003720  005761  0000000         1$:     TST     LWINDO(R1)      ;Is job doing display windowing?
003724  001404                          BEQ     4$              ;Br if not
003726                                  OCALL   WINCHR          ;Pass character to window manager
003734  103444                          BCS     9$              ;Br if we should not display this char
                                ;
                                ;  See if there is room in the output buffer for this character
                                ;
003736                          4$:     DISABL                  ;** DISABLE **
003744  026127  0000000 000010         CMP     LOTSPC(R1),#8.  ;ENOUGH SPACE IN OUTPUT BUFFER?
003752  101003                          BHI     2$              ;BR IF PLENTY OF SPACE
003754  004737  005674'                 CALL    PCSPND          ;SUSPEND PROGRAM TILL SPACE IS AVAILABLE
003760  000766                          BR      4$              ;NOW GO TRY AGAIN
                                ;
                                ;  Insert character into output buffer
                                ;
003762  005361  0000000         2$:     DEC     LOTSPC(R1)      ;SAY LESS SPACE AVAILABLE IN BUFFER
003766  016102  0000000                 MOV     LOTNXT(R1),R2   ;GET POINTER TO FREE CELL IN BUFFER
003772                                  TTMAP                   ;MAP TO TT BUFFER AREA
004006  110022                          MOVB    R0,(R2)+        ;MOVE CHAR INTO TT BUFFER
004010                                  UNMAP
004016  020261  0000000                 CMP     R2,LOTEND(R1)   ;HAVE WE HIT THE END OF THE TT BUFFER?
004022  103402                          BLO     3$              ;BR IF NOT
004024  016102  0000000                 MOV     LOTBUF(R1),R2   ;WRAPAROUND TO FRONT OF BUFFER
004030  010261  0000000         3$:     MOV     R2,LOTNXT(R1)   ;SAVE NEW BUFFER POINTER
004034                                  ENABL                   ;** ENABLE **
                                ;
                                ;  Try to start transmission to line
                                ;
004042  004777  0000000                 CALL    @TRNSTR         ;TRY TO START TRANSMISSION TO LINE
                                ;
                                ;  Finished
                                ;
004046  012602                  9$:     MOV     (SP)+,R2
004050  000207                          RETURN
```

```
                                        .SBTTL  ESCHK  -- Check for echo suppression restart
                                ;-----------------------------------------------------------------------
                                ;  ESCHK is called from PUTCHR to determine if echo suppression
                                ;  that is currently in effect should be terminated.
                                ;
                                ;  Inputs:
                                ;   R0 = Current character being output.
                                ;   R1 = Virtual line number.
                                ;
                                ;  Outputs:
                                ;   C-flag set ==> Discard the character.
                                ;   R0 = Character to output to terminal.
                                ;
004052                          ESCHK:
                                ;
                                ;  Jump to appropriate restart routine based on echo suppression class
                                ;
004052  000171  0000000                 JMP     @LESRTN(R1)     ;Jump to echo suppression restart routine
                                ;
                                ;  Restart after a normal character
                                ;
004056  010046                  ESUAC:  MOV     R0,-(SP)        ;PUSH CHARACTER BEING SENT
004060  020027  000141                  CMP     R0,#141         ;IS IT A LOWER CASE LETTER?
004064  103405                          BLO     1$              ;BR IF NOT
004066  020027  000172                  CMP     R0,#172
004072  101002                          BHI     1$
004074  162716  000040                  SUB     #40,(SP)        ;CONVERT LOWER-CASE TO UPPER-CASE
004100  122661  0000000         1$:     CMPB    (SP)+,LESCHR(R1);IS CHAR BEING SENT SAME AS ECHO-SUPPRESSION CHAR?
004104  001445                          BEQ     ESRS2           ;IF YES THEN RESTART WITH NEXT CHAR
004106  000453                          BR      ESRS1           ;IF NOT RESTART IMMEDIATELY
                                ;
                                ;  Restart output after control type activation chars (ctrl-x etc.)
                                ;
004110  120061  0000000         ESCTL:  CMPB    R0,LESCHR(R1)   ;CONTROL CHAR ECHOED BACK?
004114  001441                          BEQ     ESRS2           ;IF YES RESTART WITH NEXT CHAR
004116  120027  000136                  CMPB    R0,#'^          ;ECHO ^-CHAR?
004122  001441                          BEQ     ESRTN           ;IF YES WAIT FOR NEXT CHAR
004124  126127  0000000 000136          CMPB    LSNDCH(R1),#'^  ;WAS PREVIOUS CHAR ^?
004132  001432                          BEQ     ESRS2           ;IF YES, RESTART WITH NEXT CHAR
004134  000440                          BR      ESRS1           ;RESTART WITH THIS CHAR
                                ;
                                ;  Restart output after cr or lf.
                                ;
004136  120027  0000000         ESCRLF: CMPB    R0,#CR          ;IS THIS CHAR CR?
004142  001431                          BEQ     ESRTN           ;YES -- KEEP SKIPPING
004144  120027  0000000                 CMPB    R0,#LF          ;IS IT LF?
004150  001423                          BEQ     ESRS2           ;YES -- RESTART WITH NEXT CHAR.
004152  000431                          BR      ESRS1           ;NOT CR OF LF.  RESTART IMMEDIATELY
                                ;
                                ;  Restart after form feed
                                ;
004154  120027  0000000         ESFF:   CMPB    R0,#FF          ;ECHO FORM FEED CHAR?
004160  001417                          BEQ     ESRS2           ;IF YES, RESTART WITH NEXT CHAR
004162  120027  0000000                 CMPB    R0,#LF          ;IGNORE A STRING OF LF'S
004166  001417                          BEQ     ESRTN
004170  000422                          BR      ESRS1
                                ;
```

```
                                        ;  Restart after escape.
                                        ;
004172  120027  000044          ESESC:  CMPB    R0,#'$          ;OK TO ECHO $ TOO.
004176  001410                          BEQ     ESRS2
004200  000416                          BR      ESRS1           ;RESTART IMMEDIATELY.
                                        ;
                                        ;  Restart after tab.
                                        ;
004202  120027  000000G         ESHT:   CMPB    R0,#TAB         ;ECHO TAB CHAR?
004206  001404                          BEQ     ESRS2           ;IF YES, RESTART WITH NEXT CHAR
004210  120027  000040                  CMPB    R0,#'           ;IGNORE A STRING OF BLANKS
004214  001404                          BEQ     ESRTN
004216  000407                          BR      ESRS1
                                        ;
                                        ;  Set output restart with next character.
                                        ;
004220  042761  000000G 000000G ESRS2:  BIC     #$NOOUT,LSW3(R1);RESTART OUTPUT WITH NEXT CHAR
004226  110061  000000G         ESRTN:  MOVB    R0,LSNDCH(R1)   ;REMEMBER LAST CHAR SENT
004232  000261                          SEC                     ;SAY TO DISCARD THIS CHARACTER
004234  000404                          BR      ESXIT
                                        ;
                                        ;  Restart output immediately with this character.
                                        ;
004236  042761  000000G 000000G ESRS1:  BIC     #$NOOUT,LSW3(R1);SAY ECHO SUPPRESSION IS TURNED OFF
004244  000241                          CLC                     ;Say to restart immediately
004246  000207                  ESXIT:  RETURN
```

```
                                        .SBTTL  LIFUN  -- Process lead-in function sequences
                                ;-----------------------------------------------------------------------
                                ;  LIFUN is called from PUTCHR to process TSX lead-in character sequences
                                ;
                                ;  Inputs:
                                ;   R0 = Current character
                                ;   R1 = Virtual line number
004250  010246                  LIFUN:  MOV     R2,-(SP)
004252  010346                          MOV     R3,-(SP)
004254  010446                          MOV     R4,-(SP)
004256  010546                          MOV     R5,-(SP)
                                ;
                                ;  Enter correct processing routine
                                ;
004260  016102  000000                  MOV     LTSCMD(R1),R2   ;Get address of correct processing routine
004264  005061  000000                  CLR     LTSCMD(R1)      ;Say no processing routine pending now
004270  004712                          CALL    (R2)            ;Enter processing routine
                                ;
                                ;  Finished
                                ;
004272  012605                          MOV     (SP)+,R5
004274  012604                          MOV     (SP)+,R4
004276  012603                          MOV     (SP)+,R3
004300  012602                          MOV     (SP)+,R2
004302  000207                          RETURN
                                ;
                                ;-----------------------------------------------------------------------
                                ;  Get letter after leadin character and identify command.
                                ;
004304  162700  000101          GTCM1:  SUB     #'A,R0          ;CONVERT LETTER TO TABLE INDEX
004310  100406                          BMI     9$              ;BR IF NOT LEGAL COMMAND
004312  020027  000032                  CMP     R0,#MAXCC       ;SEE IF IN LEGAL RANGE
004316  103003                          BHIS    9$              ;BR IF TOO BIG
004320  006300                          ASL     R0              ;MAKE INTO WORD TABLE INDEX
004322  004770  004330'                  CALL    @CCVECT(R0)     ;ENTER PROCESSING ROUTINE
                                ;
                                ;  Finished
                                ;
004326  000207                  9$:     RETURN
```

```
                                ;-----------------------------------------------------------------------------
                                ;  Define TSX command control characters
                                ;
004330  004414'         CCVECT: .WORD   CMDA            ;SET RUBOUT FILLER CHARACTER
004332  004432'                 .WORD   CMDB            ;ENABLE VT50 ESC-LETTER ACTIVATION
004334  004442'                 .WORD   CMDC            ;DISABLE VT50 ESC-LETTER ACTIVATION
004336  004452'                 .WORD   CMDD            ;DEFINE ACTIVATION CHARACTER
004340  004702'                 .WORD   CMDE            ;TURN ON ECHO
004342  004712'                 .WORD   CMDF            ;TURN OFF ECHO
004344  004722'                 .WORD   CMDG            ;NOP
004346  004724'                 .WORD   CMDH            ;DISABLE VIRTUAL LINES
004350  004734'                 .WORD   CMDI            ;ENABLE LOWER CASE INPUT
004352  004764'                 .WORD   CMDJ            ;DISABLE LOWER CASE INPUT
004354  004774'                 .WORD   CMDK            ;SET DEFERRED CHAR ECHO MODE
004356  005004'                 .WORD   CMDL            ;SET IMMEDIATE CHAR ECHO MODE
004360  005022'                 .WORD   CMDM            ;SET TRANSPARENCY MODE ON
004362  005032'                 .WORD   CMDN            ;SUSPEND COMMAND FILE INPUT
004364  005052'                 .WORD   CMDO            ;RESTART COMMAND FILE INPUT
004366  005072'                 .WORD   CMDP            ;RESET USER ACTIVATION CHAR
004370  005202'                 .WORD   CMDQ            ;SET ACTIVATION ON FIELD WIDTH
004372  005356'                 .WORD   CMDR            ;TURN ON HIGH-EFFICIENCY TTY MODE
004374  005414'                 .WORD   CMDS            ;TURN ON SINGLE CHARACTER ACTIVATION MODE
004376  005424'                 .WORD   CMDT            ;TURN OFF SINGLE CHARACTER ACTIVATION MODE
004400  005434'                 .WORD   CMDU            ;ENABLE NON-WAIT TT INPUT
004402  005444'                 .WORD   CMDV            ;SET FIELD WIDTH LIMIT
004404  005634'                 .WORD   CMDW            ;TURN TAPE MODE ON
004406  005644'                 .WORD   CMDX            ;TURN TAPE MODE OFF
004410  005654'                 .WORD   CMDY            ;DISABLE AUTO LF ECHO
004412  005664'                 .WORD   CMDZ            ;ENABLE AUTO LF ECHO
000032                  MAXCC   =       <.-CCVECT>/2    ;# COMMANDS
```

```
                                        ;-----------------------------------------------------------------------------
                                        ;  Command - A
                                        ;  The next char will be the rubout filler character.
                                        ;
004414  012761  004424' 000000G CMDA:    MOV     #SETRBF,LTSCMD(R1);SET NEXT PROCESSING ROUTINE
004422  000207                           RETURN
                                        ;
                                        ;  THIS CHAR IS THE RUBOUT FILLER CHARACTER.
                                        ;
004424  110061  000000G          SETRBF: MOVB    R0,LRBFIL(R1)   ;SET NEW RUBOUT FILLER CHARACTER
004430  000207                           RETURN

                                        ;-----------------------------------------------------------------------------
                                        ;  Command - B
                                        ;  Enable activation on vt50 escape-letter sequence.
                                        ;
004432  052761  000000G 000000G CMDB:    BIS     #$VTESC,LSW5(R1);ENABLE THAT ACTIVATION SEQUENCE
004440  000207                           RETURN

                                        ;-----------------------------------------------------------------------------
                                        ;  Command - C
                                        ;  Disable activation on vt50 escape-letter sequence.
                                        ;
004442  042761  000000G 000000G CMDC:    BIC     #$VTESC,LSW5(R1);DISABLE ACTIVATION CLASS
004450  000207                           RETURN
```

```
                                        ;-----------------------------------------------------------------------------
                                        ;  Command - D
                                        ;  Define additional activation character (which follows).
                                        ;
004452  012761  004462' 000000G CMDD:    MOV     #GTSPAC,LTSCMD(R1);SET PROCESSING ROUTINE
004460  000207                           RETURN
                                        ;
                                        ;  This character is a new activation character.
                                        ;
004462  010346                  GTSPAC: MOV     R3,-(SP)
004464  010446                          MOV     R4,-(SP)
004466  010546                          MOV     R5,-(SP)
004470  032761  000000G 000000G         BIT     #$DETCH,LSW(R1) ;Is this a detached job?
004476  001075                          BNE     9$              ;Br if yes -- Ignore function
004500  004737  005120'                 CALL    DELUAC          ;DEL CHAR IF ALREADY IN TABLE
004504  016102  000000G                 MOV     LNSPAC(R1),R2   ;GET # OF CHARS DEFINED SO FAR
004510  020227  000000G                 CMP     R2,#MXSPAC      ;DEFINED ALL ALLOWED?
004514  103066                          BHIS    9$              ;CAN'T HAVE ANY MORE
004516  066102  000000G                 ADD     LSPACT(R1),R2   ;POINT TO FREE SPOT IN TABLE
004522  110012                          MOVB    R0,(R2)         ;STORE AWAY CHARACTER
004524  005261  000000G                 INC     LNSPAC(R1)      ;SAY 1 MORE CHAR IN TABLE
004530  120027  000000G                 CMPB    R0,#CTRLC       ;IS CTRL-C A SPECIAL ACTIV CHAR?
004534  001003                          BNE     1$              ;BRANCH IF NOT CTRL-C
004536  052761  000000G 000000G         BIS     #$UCTLC,LSW4(R1);REMEMBER SPECIAL CTRL-C
                                        ;
                                        ;  If user typed ahead any of the characters that are being declared
                                        ;  to be an activation char, mark them as activation chars.
                                        ;
004544  005003                  1$:     CLR     R3              ;Count chars in R3
004546  010005                          MOV     R0,R5           ;Carry new activation char in R5
004550  016102  000000G                 MOV     LINPNT(R1),R2   ;Get pointer to next char in TT buffer
004554  020361  000000G         2$:     CMP     R3,LINCNT(R1)   ;Any more chars to check?
004560  103044                          BHIS    9$              ;Br if not
004562  010204                          MOV     R2,R4           ;Save current character pointer
004564  004737  016312'                 CALL    FETCHR          ;Fetch a char from TT input buffer
004570  005203                          INC     R3              ;Count characters that we check
004572  120005                          CMPB    R0,R5           ;Is this the activation char?
004574  001367                          BNE     2$              ;Loop if not
                                        ;
                                        ;  We found the activation char in the type-ahead characters.
                                        ;  Mark it as an activation character.
                                        ;
004576  032700  000000G         4$:     BIT     #ACFLAG,R0      ;Is char already marked for activation?
004602  001013                          BNE     3$              ;Br if yes
004604  052700  000000G                 BIS     #ACFLAG,R0      ;Set activation-character flag
004610  010402                          MOV     R4,R2           ;Get back pointer to character
004612  004737  016454'                 CALL    INSCHR          ;Insert character in buffer
004616  005261  000000G                 INC     LACTIV(R1)      ;Say another pending activation char
004622  005061  000000G                 CLR     LAFSIZ(R1)      ;Clear field width activation
004626  005061  000000G                 CLR     LFWLIM(R1)      ;Clear field width limit
                                        ;
                                        ;  If the activation character is a carriage-return, check the following
                                        ;  character and delete it if it is a line-feed.
                                        ;
004632  120027  000000G         3$:     CMPB    R0,#CR          ;Is activation character a carriage-return?
004636  001346                          BNE     2$              ;Br if not
004640  020361  000000G                 CMP     R3,LINCNT(R1)   ;Any chars left in buffer?
```

```
     58 004644  103012                        BHIS    9$              ;Br if not
     59 004646  010204                        MOV     R2,R4           ;Save position of possible line feed
     60 004650  004737  016312'                CALL    FETCHR          ;Fetch character that follows CR
     61 004654  120027  0000000               CMPB    R0,#LF          ;Is that character a line feed?
     62 004660  001335                        BNE     2$              ;Br if not
     63 004662  010402                        MOV     R4,R2           ;Get back pointer to line feed
     64 004664  004737  016604'                CALL    DELCHR          ;Delete the line feed
     65 004670  000731                        BR      2$
     66                              ;
     67                              ;  Finished
     68                              ;
     69 004672  012605                  9$:   MOV     (SP)+,R5
     70 004674  012604                        MOV     (SP)+,R4
     71 004676  012603                        MOV     (SP)+,R3
     72 004700  000207                        RETURN
```

```
                                        ;-----------------------------------------------------------------------------
                                        ;  Command - E
                                        ;  Turn on character echoing.
                                        ;
004702  052761  000000G 000000G CMDE:   BIS     #$ECHO,LSW2(R1) ;ENABLE ECHOING
004710  000207                        RETURN

                                        ;-----------------------------------------------------------------------------
                                        ;  Command - F
                                        ;  Turn off character echoing.
                                        ;
004712  042761  000000G 000000G CMDF:   BIC     #$ECHO,LSW2(R1) ;DISABLE ECHOING
004720  000207                        RETURN

                                        ;-----------------------------------------------------------------------------
                                        ;  Command - G
                                        ;
004722  000207                        CMDG:   RETURN

                                        ;-----------------------------------------------------------------------------
                                        ;  Command - H
                                        ;  Disable virtual lines.
                                        ;
004724  052761  000000G 000000G CMDH:   BIS     #$NOVLN,LSW2(R1);DISABLE VIRTUAL LINES
004732  000207                        RETURN

                                        ;-----------------------------------------------------------------------------
                                        ;  Command - I
                                        ;  Enable lower case input.
                                        ;
004734  052761  000000G 000000G CMDI:   BIS     #$LC,LSW2(R1)   ;DO "SET TTY LC"
004742  106537  000000G                MFPD    @#JSWLOC        ;GET USER'S JSW VALUE
004746  052716  000000G                BIS     #LCBIT,(SP)     ;SET LOWER-CASE ENABLE FLAG
004752  011661  000000G                MOV     (SP),LJSW(R1)   ;SAVE JSW IN INTERNAL TABLE
004756  106637  000000G                MTPD    @#JSWLOC        ;STORE BACK INTO USER'S AREA
004762  000207                        RETURN

                                        ;-----------------------------------------------------------------------------
                                        ;  Command - J
                                        ;  Disable lower case input.
                                        ;
004764  042761  000000G 000000G CMDJ:   BIC     #$LC,LSW2(R1)   ;DO "SET TTY NOLC"
004772  000207                        RETURN

                                        ;-----------------------------------------------------------------------------
                                        ;  Command - K
                                        ;  Set deferred character echo mode.
                                        ;
004774  052761  000000G 000000G CMDK:   BIS     #$DEFER,LSW2(R1);SET DEFERRED ECHO FLAG
005002  000207                        RETURN

                                        ;-----------------------------------------------------------------------------
                                        ;  Command - L
                                        ;  Set immediate mode char echo.
                                        ;
005004  042761  000000G 000000G CMDL:   BIC     #$DEFER,LSW2(R1) ;Echo immediately from now on
005012  042761  000000C 000000G        BIC     #<$DODFR!$GCECO>,LSW3(R1) ;Say we are not now deferring
```

```
                                RETURN

                        ;-------------------------------------------------------------------------
                        ;  Command - M
                        ;  Set output transparency mode
                        ;
005022  052761  0000000 0000000 CMDM:   BIS     #$TRNSP,LSW3(R1);TURN ON TRANSPARENCY MODE
005030  000207                          RETURN

                        ;-------------------------------------------------------------------------
                        ;  Command - N
                        ;  Suspend input from a control file
                        ;
005032  013702  0000000         CMDN:   MOV     CFPNT,R2        ;IS A COMMAND FILE ACTIVE NOW?
005036  001404                          BEQ     1$              ;BR IF NOT
005040  010237  0000000                 MOV     R2,CFSPND       ;SAVE POINTER
005044  004737  010016'                 CALL    CFSTOP          ;Suspend command file input
005050  000207                  1$:     RETURN

                        ;-------------------------------------------------------------------------
                        ;  Command - O
                        ;  Restart input from suspended command file
                        ;
005052  013700  0000000         CMDO:   MOV     CFSPND,R0       ;IS COMMAND FILE SUSPENDED?
005056  001404                          BEQ     1$              ;BR IF NOT
005060  004737  010042'                 CALL    CFSTRT          ;Restart command file input
005064  005037  0000000                 CLR     CFSPND
005070  000207                  1$:     RETURN
```

```
                                ;-----------------------------------------------------------------------
                                ;  Command - P
                                ;  Reset user specified activation char.
                                ;
005072  012761  005102' 000000G CMDP:   MOV     #RSSPAC,LTSCMD(R1);SET PROCESSING ROUTINE
005100  000207                          RETURN
                                ;
                                ;  THIS CHAR IS THE CHAR TO RESET.
                                ;
005102  032761  000000G 000000G RSSPAC: BIT     #$DETCH,LSW(R1) ;Is this a detached job?
005110  001002                          BNE     9$              ;Br if yes -- Ignore function
005112  004737  005120'                 CALL    DELUAC          ;CALL SUBROUTINE TO DEL CHAR FROM TABLE
005116  000207                  9$:     RETURN
                                ;
                                ;  SUBROUTINE TO DELETE A USER-DEFINED ACTIVATION FROM THE TABLE
                                ;  OF ACTIVATION CHARACTERS.
                                ;  WHEN CALLED THE CHARACTER TO BE DELETED MUST BE IN R0.
                                ;
005120  010246                  DELUAC: MOV     R2,-(SP)
005122  010346                          MOV     R3,-(SP)
005124  016102  000000G                 MOV     LNSPAC(R1),R2   ;GET # OF USER ACTIVATION CHARS
005130  001421                          BEQ     3$              ;BR IF NONE TO RESET
005132  016103  000000G                 MOV     LSPACT(R1),R3   ;POINT TO START OF TABLE
005136  120023                  2$:     CMPB    R0,(R3)+        ;SEARCH FOR CHAR
005140  001402                          BEQ     1$              ;BR WHEN FOUND
005142  077203                          SOB     R2,2$           ;LOOP IF MORE TO CHECK
                                ;  COULDN'T FIND CHAR
005144  000413                          BR      3$
                                ;  FOUND CHAR
005146  112363  177776          1$:     MOVB    (R3)+,-2(R3)    ;MOVE BACK REST OF CHARS
005152  077203                          SOB     R2,1$
005154  005361  000000G                 DEC     LNSPAC(R1)      ;SAY ONE LESS CHAR TO ACTIVATE ON
005160  120027  000000G                 CMPB    R0,#CTRLC       ;DE-ACTIVATING CTRL-C?
005164  001003                          BNE     3$              ;BR IF NOT
005166  042761  000000G 000000G         BIC     #$UCTLC,LSW4(R1);REMEMBER NO MORE CTRL-C
005174  012603                  3$:     MOV     (SP)+,R3
005176  012602                          MOV     (SP)+,R2
005200  000207                          RETURN
```

```
                                ;-----------------------------------------------------------------------
                                ;  Command - Q
                                ;  Set field width as an activation condition
                                ;
005202  012761  005212' 000000G CMDQ:   MOV     #SFWAC,LTSCMD(R1);SET PROCESSING ROUTINE
005210  000207                         RETURN
                                ;
                                ;  Interpret this char as value of field width
                                ;
005212  010246                  SFWAC:  MOV     R2,-(SP)
005214  010346                          MOV     R3,-(SP)
005216  010446                          MOV     R4,-(SP)
005220  032761  000000G 000000G         BIT     #$DETCH,LSW(R1) ;Is this a detached job?
005226  001047                          BNE     9$              ;Br if yes -- Ignore function
005230  010003                          MOV     R0,R3           ;CARRY FIELD WIDTH IN R3
005232  004737  007760'                 CALL    CFTEST          ;INPUT COMING FROM COMMAND FILE?
005236  103043                          BCC     9$              ;BR IF YES
005240  004737  017254'                 CALL    SLCHK           ;IS SINGLE LINE EDITOR IN CONTROL?
005244  103436                          BCS     1$              ;BR IF YES -- SET VALUE AND LET SL HANDLE IT
005246  005761  000000G                 TST     LACTIV(R1)      ;IS ACTIVATION CHAR PENDING NOW?
005252  001035                          BNE     9$              ;IF YES THEN IGNORE FIELD WIDTH
                                ;
                                ;  No activation chars are pending and input is not
                                ;  coming from a command file.
                                ;  See if he has typed ahead enough to cause activation
                                ;
005254  005703                          TST     R3              ;IS HE RESETTING FIELD WIDTH ACTIVATION?
005256  001431                          BEQ     1$              ;BR IF YES
005260  020361  000000G                 CMP     R3,LINCNT(R1)   ;REACHED ACTIVATION ALREADY?
005264  101026                          BHI     1$              ;BR IF NOT
                                ;
                                ;  User has typed ahead enough to fill this field.
                                ;  Set activation flag in last char in field.
                                ;
005266  016102  000000G                 MOV     LINPNT(R1),R2   ;POINT TO 1ST CHAR PENDING
005272  010204                  3$:     MOV     R2,R4           ;SAVE POINTER TO NEXT CHAR
005274  004737  016312'                 CALL    FETCHR          ;FETCH THE NEXT INPUT CHARACTER
005300  103422                          BCS     9$              ;STRANGE -- NO MORE CHARS TO GET
005302  032700  000000G                 BIT     #ACFLAG,R0      ;IS THIS CHAR ALREADY FLAGGED FOR ACTIVATION?
005306  001017                          BNE     9$              ;BR IF YES
005310  077310                          SOB     R3,3$           ;LOOP TILL WE GET LAST CHAR IN FIELD
005312  052700  000000G                 BIS     #ACFLAG,R0      ;SET ACTIVATION-CHARACTER FLAG FOR CHAR
005316  010402                          MOV     R4,R2           ;GET BACK POINTER TO CHAR POS IN BUFFER
005320  004737  016454'                 CALL    INSCHR          ;REINSERT CHARACTER INTO BUFFER
005324  005261  000000G                 INC     LACTIV(R1)      ;SAY ONE MORE PENDING ACTIVATION CHAR
005330  005061  000000G                 CLR     LAFSIZ(R1)      ;SAY FIELD WIDTH ACTIVATION NOT IN EFFECT
005334  005061  000000G                 CLR     LFWLIM(R1)      ;SAY NO FIELD WIDTH LIMIT
005340  000402                          BR      9$
                                ;
                                ;  User has not typed ahead entire field.
                                ;  Remember field size as activation condition.
                                ;
005342  010061  000000G          1$:    MOV     R0,LAFSIZ(R1)   ;REMEMBER FIELD SIZE
                                ;
                                ;  Finished
                                ;
005346  012604                   9$:    MOV     (SP)+,R4
```

```
      58 005350  012603                          MOV     (SP)+,R3
      59 005352  012602                          MOV     (SP)+,R2
      60 005354  000207                          RETURN
```

```
                                        ;-----------------------------------------------------------------------
                                        ;  Command - R
                                        ;  Turn on high-efficiency tty mode
                                        ;
005356  032761  000000G 000000G CMDR:   BIT     #$DETCH,LSW(R1) ;Is this a detached job?
005364  001002                          BNE     9$              ;Br if yes -- Ignore function
005366  004737  005374'                 CALL    HION            ;TURN ON HIGH-EFFICIENCY MODE
005372  000207                  9$:     RETURN
                                        ;
                                        ;  HION IS CALLED TO ENABLE HIGH-EFFICIENCY TTY MODE.
                                        ;  IT IS CALLED FROM TSEMT AS WELL AS FROM TSEXEC.
                                        ;  WHEN CALLED, THE LINE INDEX # MUST BE IN R1.
                                        ;
005374  032761  000000G 000000G HION:   BIT     #$DETCH,LSW(R1) ;IS THIS A DETACHED JOB?
005402  001003                          BNE     1$              ;BR IF YES (CAN'T USE THIS MODE)
005404  052761  000000G 000000G         BIS     #$HITTY,LSW4(R1);ENABLE HIGH-EFFICIENCY MODE
005412  000207                  1$:     RETURN

                                        ;-----------------------------------------------------------------------
                                        ;  Command - S
                                        ;  Turn on single character activation mode
                                        ;
005414  052761  000000G 000000G CMDS:   BIS     #$CHACT,LSW5(R1);TURN ON SINGLE CHARACTER ACTIVATION
005422  000207                          RETURN

                                        ;-----------------------------------------------------------------------
                                        ;  Command - T
                                        ;  Turn off single character activation mode
                                        ;
005424  042761  000000G 000000G CMDT:   BIC     #$CHACT,LSW5(R1);TURN OFF SINGLE CHARACTER ACTIVATION
005432  000207                          RETURN

                                        ;-----------------------------------------------------------------------
                                        ;  Command - U
                                        ;  Enable non-wait TT input  (.TTINR)
                                        ;
005434  052761  000000G 000000G CMDU:   BIS     #$NOWTT,LSW5(R1);ENABLE NON-WAIT TT INPUT
005442  000207                          RETURN
```

```
                                ;-----------------------------------------------------------------------
                                ;  Command - V
                                ;  Set field width limit.
                                ;
005444  012761  005454' 000000G CMDV:   MOV     #SFWL,LTSCMD(R1);NEXT CHAR WILL BE FIELD WIDTH SIZE
005452  000207                        RETURN
                                ;
                                ;  This character specifies the field size
                                ;
005454  010346                  SFWL:   MOV     R3,-(SP)
005456  010446                          MOV     R4,-(SP)
005460  010546                          MOV     R5,-(SP)
005462  032761  000000G 000000G        BIT     #$DETCH,LSW(R1) ;Is this a detached job?
005470  001055                        BNE     9$              ;Br if yes -- Ignore function
005472  005061  000000G               CLR     LFWLIM(R1)      ;Initially clear field width
005476  010004                        MOV     R0,R4           ;IS SPECIFIED FIELD WIDTH ZERO?
005500  001451                        BEQ     9$              ;Br if yes
005502  004737  007760'          1$:  CALL    CFTEST          ;IS INPUT COMING FROM A COMMAND FILE?
005506  103046                        BCC     9$              ;BR IF YES
005510  004737  017254'               CALL    SLCHK           ;IS SINGLE LINE EDITOR RUNNING?
005514  103441                        BCS     6$              ;BR IF YES
                                ;
                                ;  See if user typed ahead.
                                ;  Check size of field and discard any chars beyond end of specified size.
                                ;
005516  005003                  14$:  CLR     R3              ;COUNT TOTAL CHARACTERS IN R3
005520  016102  000000G               MOV     LINPNT(R1),R2   ;GET POINTER TO NEXT CHAR IN BUFFER
005524  020361  000000G          5$:  CMP     R3,LINCNT(R1)   ;HAVE WE CHECKED ALL CHARS IN TT BUFFER?
005530  103033                        BHIS    6$              ;BR IF YES -- NO OVERFLOW HAS OCCURRED YET
005532  010205                        MOV     R2,R5           ;SAVE POINTER TO NEXT CHAR
005534  004737  016312'               CALL    FETCHR          ;GET NEXT CHAR FROM TT INPUT BUFFER
005540  120027  000000G               CMPB    R0,#CR          ;IS THIS CHAR CARRIAGE-RETURN?
005544  001427                        BEQ     9$              ;BR IF YES -- THIS ENDS THE FIELD
005546  032700  000000G               BIT     #ACFLAG,R0      ;IS THIS CHAR AN ACTIVATION CHAR?
005552  001024                        BNE     9$              ;BR IF YES -- FIELD TERMINATED WITHOUT OVRFLOW
005554  120027  000000G               CMPB    R0,#ESCFLG      ;IS THIS CHAR PART OF VT100 ESCAPE SEQ?
005560  001004                        BNE     2$              ;BR IF NOT
005562  032761  000000G 000000G        BIT     #$VTESC,LSW5(R1);ARE WE ACTIVATING ON VT100 ESC SEQUENCES?
005570  001015                        BNE     9$              ;BR IF YES -- FIELD WIDTH OK
005572  005203                  2$:   INC     R3              ;COUNT TOTAL # CHARS IN FIELD
005574  020304                        CMP     R3,R4           ;DOES THIS CHAR OVERFLOW THE FIELD?
005576  101752                        BLOS    5$              ;BR IF NOT
                                ;
                                ;  Field overflow.  Discard all chars typed beyond end of field.
                                ;
005600  010502                        MOV     R5,R2           ;POINT TO CHAR THAT OVERFLOWED FIELD
005602  004737  016604'          10$: CALL    DELCHR          ;DELETE ALL CHARS THAT ARE BEYOND FIELD END
005606  103375                        BCC     10$             ;LOOP IF MORE CHARS TO DELETE
                                ;  Ring bell to signal user that field overflowed.
005610  012700  000000G               MOV     #BELL,R0        ;GET BELL CHARACTER
005614  004737  003462'               CALL    QUECHR          ;END TO TERMINAL
                                ;
                                ;  Field did not overflow
                                ;
005620  010461  000000G          6$:  MOV     R4,LFWLIM(R1)   ;SET FIELD WIDTH LIMIT FOR THIS FIELD
                                ;
                                ;  Finished
```

```
      58                                        ;
      59 005624  012605                  9$:   MOV     (SP)+,R5
      60 005626  012604                        MOV     (SP)+,R4
      61 005630  012603                        MOV     (SP)+,R3
      62 005632  000207                        RETURN
```

```
                                        ;-----------------------------------------------------------------------------
                                        ;  Command - W
                                        ;  Turn "paper-tape" mode on.
                                        ;
005634  052761  0000000 0000000 CMDW:   BIS     #$TAPE,LSW2(R1) ;TURN TAPE MODE ON
005642  000207                          RETURN

                                        ;-----------------------------------------------------------------------------
                                        ;  Command - X
                                        ;  Turn "paper-tape" mode off.
                                        ;
005644  042761  0000000 0000000 CMDX:   BIC     #$TAPE,LSW2(R1) ;TURN TAPE MODE OFF
005652  000207                          RETURN

                                        ;-----------------------------------------------------------------------------
                                        ;  Command - Y
                                        ;  Disable echoing of line-feed following carriage-return
                                        ;
005654  052761  0000000 0000000 CMDY:   BIS     #$NOLF,LSW6(R1) ;DISABLE AUTO LINE-FEED ECHO
005662  000207                          RETURN

                                        ;-----------------------------------------------------------------------------
                                        ;  Command - Z
                                        ;  Enable echoing of line-feed following carriage-return
                                        ;
005664  042761  0000000 0000000 CMDZ:   BIC     #$NOLF,LSW6(R1) ;REENABLE AUTO LINE-FEED ECHO
005672  000207                          RETURN
```

```
                                ;-----------------------------------------------------------------------
                                ;  PCSPND is called to suspend execution of the current job because
                                ;  the terminal output buffer is full.
                                ;
                                ;  Inputs:
                                ;   R1 = Job index number.
                                ;
                                ;  Outputs:
                                ;   All registers are preserved.
                                ;
005674  010046                  PCSPND: MOV     R0,-(SP)
005676  004737  017466'                 CALL    SIGWAT          ;SIGNAL JOB WAIT
005702  012700  0000000                 MOV     #S$OTWT,R0      ;CHANGE JOB STATE TO WAITING-FOR-OUTPUT-SPACE
005706  004737  0000000                 CALL    QHDSPN          ;CHANGE STATE AND SUSPEND THE JOB
005712  004737  0000000                 CALL    CHKABT          ;SEE IF WE WERE ABORTED WHILE ASLEEP
005716  012600                          MOV     (SP)+,R0
005720  000207                          RETURN
                                ;-----------------------------------------------------------------------
                                ;  TRYCHR is similar to PUTCHR except if the user's
                                ;  output buffer is full TRYCHR simply returns
                                ;  and does not halt execution.
                                ;  TRYCHR also does not give special treatment to
                                ;  such chars as form feed, backspace, etc.
                                ;  when called r0 must contain the character to be
                                ;  sent and r1 must contain the virtual line index #.
                                ;  all registers are preserved.
                                ;
005722                          TRYCHR: DISABL                  ;** DISABLE **
005730  005761  0000000                 TST     LOTSPC(R1)      ;IS THERE ROOM FOR THE CHAR?
005734  001402                          BEQ     9$              ;BR IF NOT
005736  004737  003462'                 CALL    QUECHR          ;PUT CHARACTER INTO BUFFER
005742  000207                  9$:     RETURN
```

```
      1                                 ;-----------------------------------------------------
      2                                 ;  OTREGO IS CALLED TO RESTART A USER WHO IS SUSPENDED
      3                                 ;  WAITING FOR SPACE IN THE OUTPUT BUFFER.
      4                                 ;  WHEN CALLED THE USER NUMBER MUST BE IN R1.
      5                                 ;
      6 005744  012700  0000000         OTREGO: MOV     #S$OTFN,R0      ;PUT USER IN OUTPUT-NEEDED QUEUE
      7 005750  004737  0000000                 CALL    ENQTL
      8 005754  000207                          RETURN
```

```
                                        .SBTTL
                                        .SBTTL  **  Program Level Input Character Processing  **
                                        .SBTTL  GETCHR -- Get next input char
                                ;-----------------------------------------------------------------
                                ;  GETCHR -- ROUTINE TO MOVE THE NEXT CHARACTER FROM THE
                                ;  INPUT BUFFER TO R0.
                                ;  IF NO CHARACTERS ARE AVAILABLE THE USER IS SUSPENDED
                                ;  UNTIL A RECORD IS RECEIVED.
                                ;
005756  010146                  GETCHR: MOV     R1,-(SP)
005760  010246                          MOV     R2,-(SP)
005762  113701  0000000                 MOVB    CORUSR,R1       ;GET USER'S INDEX #
005766  042761  0000000 0000000         BIC     #$GCESC,LSW3(R1);WE'RE NOT PROCESSING VT50 ESC SEQUENCE
005774  032761  000000C 0000000 GCH1:   BIT     #<$DISCN+$CTRLC>,LSW(R1);DISCONNECT OR CTRL-C?
006002  001402                          BEQ     8$              ;BRANCH IF NEITHER
006004  004737  0000000                 CALL    STOP            ;STOP PROGRAM IF EITHER
                                ;
                                ;  See if input is coming from a command file or from the terminal.
                                ;
006010  004737  007262'         8$:     CALL    GTCFCH          ;TRY TO GET A CHARACTER FROM A COMMAND FILE
006014  103402                          BCS     15$             ;BR IF DID NOT GET ONE
006016  000137  006356'                 JMP     GCCKES          ;PROCESS CHAR FROM COMMAND FILE
                                ;
                                ;  We did not get a character from a command file.
                                ;  If we are processing a logoff command file, finish the logoff now.
                                ;
006022  032761  0000000 0000000 15$:    BIT     #$LOFCF,LSW9(R1);Are we processing a logoff command file?
006030  001405                          BEQ     13$             ;Br if not
006032  052761  0000000 0000000         BIS     #$DOOFF,LSW(R1) ;Force job logoff
006040  004737  0000000                 CALL    STOP
                                ;
                                ;  If this is a detached job, halt its execution if command file finished
                                ;
006044  032761  0000000 0000000 13$:    BIT     #$DETCH,LSW(R1) ;Is this a detached job?
006052  001405                          BEQ     16$             ;Br if not
006054  052761  0000000 0000000         BIS     #$DISCN,LSW(R1) ;Force job logoff
006062  004737  0000000                 CALL    STOP            ;Halt detached jobs that want terminal input
                                ;
                                ;  See if we should get a character from the single line editor.
                                ;
006066  032761  000000C 0000000 16$:    BIT     #<SPCTTY!DISSLE>,LJSW(R1);Special TTY mode or SL disabled?
006074  001031                          BNE     11$             ;Br if yes -- don't use SL
006076  032761  0000000 0000000         BIT     #$SLON,LSW7(R1) ;Is SL enabled for this line?
006104  001425                          BEQ     11$             ;Br if not
006106  032761  000000C 0000000         BIT     #<$ODTMD!$HITTY>,LSW4(R1);Are we in ODT or high efficiency?
006114  001021                          BNE     11$             ;Br if yes
006116  032761  0000000 0000000         BIT     #$VTESC,LSW5(R1);VTxxx activation enabled?
006124  001015                          BNE     11$             ;Br if yes
006126  032761  0000000 0000000         BIT     #$GTLIN,LSW4(R1);Is a .GTLIN being done?
006134  001004                          BNE     12$             ;Br if yes
006136  032761  0000000 0000000         BIT     #$SLTTY,LSW7(R1);Is SL enabled for TTY input?
006144  001405                          BEQ     11$             ;Br if not
006146                          12$:    OCALL   GTSLCH          ;Get character from single line editor
006154  000137  006774'                 JMP     GCEXIT          ;Exit from GETCHR
                                ;
                                ;  Get character from the terminal.
                                ;
```

```
     58 006160  032761  000000G 000000G 11$:     BIT     #$NOINT,LSW7(R1);Does user want non-interactive execution?
     59 006166  001006                          BNE     14$             ;Br if yes
     60 006170  013761  000000G 000000G          MOV     VINTIO,LHIPCT(R1);Reset interactive I/O limit for job
     61 006176  013761  000000G 000000G          MOV     VQUAN1,LITIME(R1);Reset interactive CPU time limit
     62 006204  005761  000000G          14$:    TST     LACTIV(R1)      ;Any activation chars pending?
     63 006210  001003                          BNE     9$              ;Branch if yes
     64 006212  004737  011156'                 CALL    ILWAIT          ;Wait for activation
     65 006216  000666                          BR      GCH1            ;Go try again
     66 006220  016102  000000G          9$:     MOV     LINPNT(R1),R2   ;Get pointer to next char to get
     67 006224  004737  016604'                 CALL    DELCHR          ;Get char and delete from input buffer
     68 006230  103661                          BCS     GCH1            ;Br if no more characters are available
     69 006232  005061  000000G                 CLR     LRTCHR(R1)      ;Say no read time-out pending
     70                                     ;
     71                                     ;  See if we are in single-character-activation mode.
     72                                     ;
     73 006236  032761  000000G 000000G          BIT     #SPCTTY,LJSW(R1);DOES PROGRAM WANT SINGLE-CHAR INPUT?
     74 006244  001415                          BEQ     6$              ;BR IF NOT
     75 006246  032761  000000G 000000G          BIT     #$CHACT,LSW5(R1);IS SINGLE-CHARACTER INPUT ENABLED?
     76 006254  001411                          BEQ     6$              ;BR IF NOT
     77 006256  004737  017414'                 CALL    CVTLC           ;SET IF WE NEED TO CONVERT TO UPPER-CASE
     78 006262  032700  000000G                 BIT     #ACFLAG,R0      ;IS THIS AN ACTIVATION CHARACTER?
     79 006266  001402                          BEQ     10$             ;BR IF NOT
     80 006270  000137  006526'                 JMP     GCCKDS
     81 006274  000137  006572'          10$:    JMP     GCCKCC
     82                                     ;
     83                                     ;  See if we are in high-efficiency mode.
     84                                     ;
     85 006300  032761  000000G 000000G 6$:     BIT     #$HITTY,LSW4(R1);ARE WE IN HIGH-EFFICIENCY MODE?
     86 006306  001402                          BEQ     3$              ;BR IF NOT
     87                                     ;
     88                                     ;  We are in high-efficiency mode
     89                                     ;
     90 006310  000137  006732'          5$:     JMP     GCEND           ;DO EXPRESS CHARACTER PROCESSING
     91                                     ;
     92                                     ;  SEE IF THIS IS FLAG CHAR MEANING NEXT CHAR IS PART OF
     93                                     ;  VT50 ESCAPE SEQUENCE.
     94                                     ;
     95 006314  032761  000000G 000000G 3$:     BIT     #$GCESC,LSW3(R1);IS THIS CHAR PART OF ESC SEQUENCE?
     96 006322  001402                          BEQ     1$              ;BR IF NOT
     97 006324  000137  006512'                 JMP     GCCKAC          ;TAKE CHAR WITHOUT FURTHER CHECKING
     98 006330  120027  000000G          1$:     CMPB    R0,#ESCFLG      ;IS THIS FLAG FOR NEXT CHAR IN ESC SEQ?
     99 006334  001010                          BNE     GCCKES          ;BR IF NOT
    100 006336  032761  000000G 000000G          BIT     #$VTESC,LSW5(R1);ARE WE ACTIVATING ON ESCAPE SEQUENCES?
    101 006344  001404                          BEQ     GCCKES          ;BR IF NOT
    102 006346  052761  000000G 000000G          BIS     #$GCESC,LSW3(R1);REMEMBER NEXT CHAR PART OF ESC SEQ
    103 006354  000721                          BR      9$              ;GO GET NEXT CHAR
```

```
                                        ;
                                        ;   WE HAVE THE NEXT CHARACTER IN R0.
                                        ;   SEE IF WE NEED TO TRANSLATE LOWER CASE CHARS TO UPPER CASE.
                                        ;
006356  004737  017414'         @CCKES: CALL    CVTLC           ;CONVERT LC TO UC IF NEEDED
                                        ;
                                        ;   SEE IF WE NEED TO DO ECHO SUPPRESSION.
                                        ;
006362  032761  0000000 0000000         BIT     #SPCTTY,LJSW(R1);IS ECHO SUPPRESSION NEEDED?
006370  001450                          BEQ     @CCKAC          ;BRANCH IF NOT
006372  032761  0000000 0000000         BIT     #$CHACT,LSW5(R1);ARE WE IN SINGLE-CHAR-ACTIVATION MODE?
006400  001074                          BNE     @CCKCC          ;BR IF YES
                                        ;
                                        ;   BEGIN ECHO SUPPRESSION.
                                        ;
006402  010346                          MOV     R3,-(SP)
006404  010002                          MOV     R0,R2           ;GET THE CHAR
006406  042702  177400                  BIC     #177400,R2      ;CLEAR ACTIVATION CHAR FLAG
                                        ;   SEE IF THIS IS A REGULAR OR CONTROL CHARACTER
006412  120227  000037                  CMPB    R2,#37          ;REGULAR OR CONTROL?
006416  101013                          BHI     4$              ;BRANCH IF REGULAR CHAR
                                        ;   THIS IS A CONTROL CHAR -- CHECK FOR SPECIAL CASES.
006420  012703  000004                  MOV     #NESCTL,R3      ;# OF SPECIAL CONTROL CHARS
006424  120263  007006'          5$:     CMPB    R2,SESCTL(R3)   ;IS THIS ONE?
006430  001402                          BEQ     8$              ;BRANCH IF YES
006432  005303                          DEC     R3              ;TRY REST
006434  100373                          BPL     5$
                                        ;   FALL THROUGH WITH R3=-1 FOR REGULAR CONTROL CHAR.
006436  006303                  8$:     ASL     R3              ;GET WORD TABLE INDEX
006440  016303  007016'                 MOV     SESRTN(R3),R3   ;GET ROUTINE ADDRESS
006444  000402                          BR      6$
                                        ;   THIS IS NOT A CONTROL CHAR.
006446  012703  004056'          4$:     MOV     #ESUAC,R3       ;SET NORMAL HANDLER ROUTINE
006452  010361  0000000         6$:     MOV     R3,LESRTN(R1)   ;SET HANDLER ROUTINE FOR PUTCHR
006456  120227  000141                  CMPB    R2,#141         ;IS CHAR A LOWER-CASE LETTER?
006462  103405                          BLO     1$              ;BR IF NOT
006464  120227  000172                  CMPB    R2,#172
006470  101002                          BHI     1$
006472  162702  000040                  SUB     #40,R2          ;CONVERT LOWER-CASE TO UPPER-CASE
006476  110261  0000000         1$:     MOVB    R2,LESCHR(R1)   ;REMEMBER CHAR
006502  052761  0000000 0000000         BIS     #$NOOUT,LSW3(R1);TURN ON ECHO SUPPRESSION
006510  012603                          MOV     (SP)+,R3
```

```
                                        ;
                                        ;  See if this is an activation character.
                                        ;
006512  004737  007760'         GCCKAC: CALL    CFTEST          ;INPUT FROM CONTROL FILE?
006516  103017                          BCC     GCCKCE          ;BR IF YES
006520  032700  0000000                 BIT     #ACFLAG,R0      ;IS THIS AN ACTIVATION CHAR?
006524  001474                          BEQ     GCCKDE          ;BRANCH IF NOT.
                                        ;
                                        ;  This is an activation character.
                                        ;  See if we need to start doing deferred echoing.
                                        ;
006526  032761  0000000 0000000 GCCKDS: BIT     #$DODFR,LSW3(R1);IS ECHOING BEING DEFERRED?
006534  001416                          BEQ     GCCKCC          ;BRANCH IF NOT
006536  032761  0000000 0000000         BIT     #$GCECO,LSW3(R1);HAVE WE ALREADY STARTED?
006544  001010                          BNE     GCEAC           ;BRANCH IF YES
006546  052761  0000000 0000000         BIS     #$GCECO,LSW3(R1);BEGIN DEFERRED ECHOING WITH NEXT CHAR
006554  000406                          BR      GCCKCC
                                        ;
                                        ;  See if want to list the contents of a control file.
                                        ;
006556  032761  0000000 0000000 GCCKCE: BIT     #$QUIET,LSW4(R1);LIST COMMAND FILE?
006564  001002                          BNE     GCCKCC          ;BR IF NO LIST
                                        ;
                                        ;  ECHO THE ACTIVATION CHAR NOW.
                                        ;
006566  004737  007030'         GCEAC:  CALL    GCECHO          ;ECHO THE CHARACTER
                                        ;
                                        ;  See if character we got is control-c.
                                        ;
006572  120027  0000000         GCCKCC: CMPB    R0,#CTRLC       ;IS CHAR CTRL-C?
006576  001055                          BNE     GCEND           ;BRANCH IF NOT
006600  032761  0000000 0000000         BIT     #$DBGMD,LSW6(R1);IS A DEBUGGING PROGRAM IN CONTROL?
006606  001016                          BNE     7$              ;BR IF YES
006610  032761  0000000 0000000         BIT     #$UCTLC,LSW4(R1);IS CTRL-C A USER DEF ACTIV CHAR?
006616  001045                          BNE     GCEND           ;BRANCH IF IT IS
006620  005761  0000000                 TST     LSCCA(R1)       ;DID USER DO .SCCA?
006624  001042                          BNE     GCEND           ;BR IF YES
006626  032761  0000000 0000000         BIT     #$CCLRN,LSW5(R1);IS CCL TRANSLATOR RUNNING?
006634  001403                          BEQ     7$              ;BR IF NOT
006636  004737  007760'                 CALL    CFTEST          ;IS INPUT COMING FROM A COMMAND FILE?
006642  103033                          BCC     GCEND           ;BR IF YES
006644  032737  0000000 0000000 7$:     BIT     #LF$IN,LOGFLG   ;Are we logging input characters?
006652  001417                          BEQ     8$              ;BR IF NOT
006654  004737  007760'                 CALL    CFTEST          ;IS INPUT COMING FROM A CONTROL FILE?
006660  103404                          BCS     6$              ;BR IF NOT
006662  032761  0000000 0000000         BIT     #$QUIET,LSW4(R1);SHOULD WE DISPLAY CONTROL FILES?
006670  001010                          BNE     8$              ;BR IF NOT
006672  032761  0000000 0000000 6$:     BIT     #$ECHO,LSW2(R1) ;ARE WE ECHOING CHARACTERS?
006700  001404                          BEQ     8$              ;DON'T LOG IF NOT ECHOING
006702  004737  010714'                 CALL    LOGCHR          ;LOG THE CONTROL-C
006706  004737  010764'                 CALL    LOGCR           ;LOG CR-LF
006712  004737  0000000         8$:     CALL    STOP            ;STOP PROGRAM EXECUTION
```

```
                                        ;
                                        ;   THIS CHAR IS NOT AN ACTIVATION CHAR.
                                        ;
                                        ;   SEE IF WE NEED TO DO DEFERRED ECHOING.
                                        ;
006716  032761  000000G 000000G GCCKDE: BIT     #$GCECO,LSW3(R1);HAVE WE STARTED DEFERRED ECHOING?
006724  001402                        BEQ     GCEND           ;BRANCH IF NOT
006726  004737  007030'                CALL    GCECHO          ;ECHO THE CHARACTER
                                        ;
                                        ;   See if we should write character to log file
                                        ;
006732  032737  000000G 000000G GCEND:  BIT     #LF$IN,LOGFLG   ;Are we logging input characters?
006740  001415                        BEQ     GCEXIT          ;Br if not
006742  004737  007760'                CALL    CFTEST          ;Is input coming from a control file?
006746  103404                        BCS     1$              ;Br if not
006750  032761  000000G 000000G       BIT     #$QUIET,LSW4(R1);Should we log control file characters?
006756  001006                        BNE     GCEXIT          ;Br if not
006760  032761  000000G 000000G 1$:   BIT     #$ECHO,LSW2(R1) ;Is echo suppression in effect?
006766  001402                        BEQ     GCEXIT          ;Do not log if not echoing
006770  004737  010714'                CALL    LOGCHR          ;Log the character
006774  042700  177400          GCEXIT: BIC     #^C377,R0       ;Mask character to 8 bits
007000  012602                        MOV     (SP)+,R2
007002  012601                        MOV     (SP)+,R1
007004  000207                        RETURN
```

```
                                ;
                                ;  TABLE OF CONTROL CHARACTERS WHICH REQUIRE SPECIAL
                                ;  PROCESSING WITH REGARD TO ECHO SUPPRESSION.
                                ;
                                ;  TABLE OF CONTROL CHARACTERS.
                                ;
007006      0000                SESCTL: .BYTE   FF              ;FORM FEED
007007      0000                        .BYTE   ESC             ;ESCAPE
007010      0000                        .BYTE   LF              ;LINE FEED
007011      0000                        .BYTE   CR              ;CARRIAGE RETURN
007012      0000                        .BYTE   TAB             ;TAB
            000004              NESCTL  =       .-SESCTL-1      ;# OF SPECIAL CHARS.
                                        .EVEN
                                ;
                                ;  PARALLEL TABLE OF ADDRESSES OF ROUTINES TO RESTART
                                ;  OUTPUT WHEN ECHO SUPPRESSION IS IN EFFECT FOR
                                ;  SPECIAL CONTROL CHAR.
                                ;  SESRTN TABLE MUST BE PARALLEL TO SESCTL TABLE.
                                ;  NOTE:  (-1) TABLE ENTRY IS USED FOR REGULAR CONTROL CHARS.
                                ;
007014  004110'                         .WORD   ESCTL           ;(-1) REGULAR CONTROL CHAR
007016  004154'                 SESRTN: .WORD   ESFF            ;FORM FEED
007020  004172'                         .WORD   ESESC           ;ESCAPE
007022  004136'                         .WORD   ESCRLF          ;LINE FEED
007024  004136'                         .WORD   ESCRLF          ;CARRIAGE RETURN
007026  004202'                         .WORD   ESHT            ;TAB
```

```
                                        ;-----------------------------------------------------
                                        ;  GCECHO IS CALLED TO ECHO A CHARACTER AS IT IS PASSED
                                        ;  TO THE USER IF WE ARE IN DEFERRED ECHO MODE.
                                        ;  WHEN CALLED, THE CHARACTER TO BE ECHOED MUST BE IN
                                        ;  R0 AND THE USER INDEX NUMBER MUST BE IN R1.
                                        ;  ALL REGISTERS ARE PRESERVED.
                                        ;
007030  004737  017214'          GCECHO: CALL    SCACHK          ;ARE WE IN SINGLE CHARACTER INPUT MODE?
007034  103511                          BCS     99$             ;BR IF YES -- DON'T ECHO CHARACTER
007036  032761  000000G 000000G          BIT     #$ECHO,LSW2(R1) ;IS CHAR ECHOING WANTED?
007044  001505                          BEQ     99$             ;RETURN IF NOT
007046  032761  000000G 000000G          BIT     #$GCESC,LSW3(R1);IS THIS CHAR PART OF ESC SEQUENCE?
007054  001101                          BNE     99$             ;BR IF YES
007056  010046                          MOV     R0,-(SP)
007060  010246                          MOV     R2,-(SP)
007062  010346                          MOV     R3,-(SP)
007064  042700  000000G                BIC     #ACFLAG,R0      ;STRIP OFF ACTIVATION-CHAR FLAG
                                        ;
                                        ;  See if we should echo line-feed characters
                                        ;
007070  120027  000000G                 CMPB    R0,#LF          ;IS THIS CHAR LINE-FEED?
007074  001011                          BNE     7$              ;BR IF NOT
007076  032761  000000G 000000G          BIT     #$NOLF,LSW6(R1) ;IS LINE-FEED ECHO SUPPRESSION IN EFFECT?
007104  001405                          BEQ     7$              ;BR IF NOT
007106  032761  000000G 000000G          BIT     #$DBGMD,LSW6(R1);IS A DEBUGGER RUNNING NOW?
007114  001456                          BEQ     2$              ;BR IF NOT (DON'T ECHO LF)
007116  000453                          BR      5$              ;ECHO LF
                                        ;
                                        ;  SEE IF CHAR IS A USER DEFINED ACTIVATION CHAR.
                                        ;
007120  016102  000000G          7$:    MOV     LNSPAC(R1),R2   ;GET # OF USER DEF ACTIV CHARS
007124  001405                          BEQ     1$              ;BRANCH IF NONE
007126  016103  000000G                 MOV     LSPACT(R1),R3   ;POINT TO TABLE FOR USER
007132  120023                  3$:     CMPB    R0,(R3)+        ;SEE IF THIS IS ONE
007134  001446                          BEQ     2$              ;BRANCH IF IT IS
007136  077203                          SOB     R2,3$           ;LOOP IF MORE TO CHECK
                                        ;  SEE IF CHARACTER IS ESCAPE.
007140  120027  000037          1$:     CMPB    R0,#37          ;REGULAR OR CONTROL CHAR?
007144  101040                          BHI     5$              ;BRANCH IF REGULAR CHARACTER
007146  120027  000000G                 CMPB    R0,#ESC         ;IS CHAR ESCAPE?
007152  001003                          BNE     4$              ;BRANCH IF NOT
007154  112700  000044                  MOVB    #'$,R0          ;OTHERWISE, ECHO $ FOR ESCAPE
007160  000432                          BR      5$
                                        ;  CHECK FOR SPECIAL CONTROL CHARACTERS.
007162  120027  000000G          4$:    CMPB    R0,#CTRLZ       ;CTRL-Z?
007166  001406                          BEQ     6$              ;BRANCH IF YES
007170  120027  000000G                 CMPB    R0,#CTRLC       ;CTRL-C?
007174  001403                          BEQ     6$
007176  120027  000000G                 CMPB    R0,#CTRLX       ;CTRL-X?
007202  001021                          BNE     5$              ;BRANCH IF NOT SPECIAL CONTROL CHAR
                                        ;  ECHO ^-CHAR FOR SPECIAL CONTROL CHARS
007204  110003                  6$:     MOVB    R0,R3           ;SAVE CONTROL CHAR
007206  112700  000136                  MOVB    #136,R0         ;ECHO ^
007212  004737  003056'                 CALL    PUTCH1
007216  110300                          MOVB    R3,R0           ;GET CONTROL CHAR
007220  052700  000100                  BIS     #100,R0         ;CONVERT TO PRINTING CHAR
007224  004737  003056'                 CALL    PUTCH1          ;PRINT CHAR
```

```
      58 007230  112700  000000G                 MOVB    #CR,R0          ;ECHO CR
      59 007234  004737  003056'                 CALL    PUTCH1
      60 007240  112700  000000G                 MOVB    #LF,R0
      61 007244  000400                          BR      5$
      62                                ;
      63                                ;  THIS IS A REGULAR CHARACTER.
      64                                ;
      65                                ;  ECHO CHAR IN R0.
      66 007246  004737  003056'        5$:     CALL    PUTCH1          ;ECHO THE CHARACTER
      67                                ;  RETURN
      68 007252  012603                 2$:     MOV     (SP)+,R3
      69 007254  012602                         MOV     (SP)+,R2
      70 007256  012600                         MOV     (SP)+,R0
      71 007260  000207                 99$:    RETURN
```

```
                                        .SBTTL  GTCFCH -- Try to get char from command file
                                ;-----------------------------------------------------------------------
                                ;  GTCFCH is called to try to obtain a character from a command file.
                                ;
                                ;  Inputs:
                                ;   R1 = Job index number.
                                ;
                                ;  Outputs:
                                ;   C-flag cleared if a character is obtained from command file.
                                ;   R0 = Character obtained.
                                ;
                                ;  See if we are getting characters from a command file.
                                ;
007262  004737  007760'       GTCFCH: CALL    CFTEST          ;SEE IF INPUT IS COMING FROM A COMMAND FILE
007266  103543                        BCS     13$             ;BR IF NOT
                                ;
                                ;  See if we are holding a character
                                ;
007270  113700  000000G               MOVB    CFHOLD,R0       ;ARE WE HOLDING A CHARACTER?
007274  001403                        BEQ     10$             ;BR IF NOT
007276  105037  000000G               CLRB    CFHOLD          ;SAY CHARACTER IS GONE
007302  000532                        BR      12$             ;TAKE SUCCESSFUL RETURN
                                ;
                                ;  Input is coming from a command file.
                                ;
007304  004737  007600'       10$:    CALL    CFCHAR          ;GET CHAR FROM CONTROL FILE
007310  103531                        BCS     14$             ;BR IF END OF COMMAND FILE HIT
007312  120027  000136                CMPB    R0,#'^          ;IS THIS A CONTROL CHAR?
007316  001124                        BNE     12$             ;BRANCH IF NOT
                                ;
                                ;  We found "^" character in command file.
                                ;  This means we may have to treat the next character as a control char.
                                ;
007320  004737  007600'               CALL    CFCHAR          ;GET NEXT CHAR
007324  103523                        BCS     14$             ;BR IF END OF COMMAND FILE HIT
                                ;
                                ;  ^$ is interpreted as escape
                                ;
007326  120027  000044                CMPB    R0,#'$          ;TREAT ^$ AS ESCAPE CHAR
007332  001003                        BNE     3$
007334  012700  000000G               MOV     #ESC,R0
007340  000513                        BR      12$
                                ;
                                ;  ^^ is interpreted as a single ^
                                ;
007342  120027  000136        3$:     CMPB    R0,#'^          ;ANOTHER "^"?
007346  001510                        BEQ     12$             ;IF YES THEN RETURN "^" AS CHARACTER
                                ;
                                ;  ^( means stop listing control file
                                ;
007350  120027  000050                CMPB    R0,#'(          ;STOP LISTING COMMAND?
007354  001007                        BNE     4$              ;BR IF NOT
007356  052761  000000G 000000G       BIS     #$QUIET,LSW4(R1);SET NO LISTING FLAG
007364  042761  000000G 000000G       BIC     #$CFSOT,LSW4(R1);ALLOW PROGRAM OUTPUT TO PRINT
007372  000744                        BR      10$             ;GO GET NEXT CHAR FROM FILE
                                ;
                                ;  ^) means start listing control file
```

```
                                        ;
007374  120027  000051            4$:   CMPB    R0,#'(          ;START-LISTING COMMAND?
007400  001004                          BNE     7$              ;BR IF NOT
007402  042761  000000C 000000G         BIC     #<$QUIET!$CFSOT>,LSW4(R1);START LISTING FILE
007410  000735                          BR      10$             ;GO GET NEXT CHAR
                                        ;
                                        ;  ^! means suppress all output -- command file and program
                                        ;
007412  120027  000041            7$:   CMPB    R0,#'!          ;SUPPRESS OUTPUT?
007416  001004                          BNE     5$              ;BR IF NOT
007420  052761  000000C 000000G         BIS     #<$CFSOT!$QUIET>,LSW4(R1);BEGIN SUPPRESSING OUTPUT
007426  000726                          BR      10$
                                        ;
                                        ;  ^> means accept all chars from @file
                                        ;
007430  120027  000076            5$:   CMPB    R0,#'>          ;ACCEPT ALL CHARS?
007434  001004                          BNE     6$              ;BR IF NOT
007436  052761  000000G 000000G         BIS     #$CFALL,LSW4(R1);SET FLAG
007444  000717                          BR      10$
                                        ;
                                        ;  ^< means accept only .gtlin chars from @file
                                        ;
007446  120027  000074            6$:   CMPB    R0,#'<          ;ACCEPT ONLY .GTLIN CHARS?
007452  001004                          BNE     2$              ;BR IF NOT
007454  042761  000000G 000000G         BIC     #$CFALL,LSW4(R1);RESET FLAG
007462  000677                          BR      GTCFCH
                                        ;
                                        ;  ^digit is used to indicate a parameter substitution
                                        ;
007464  120027  000061            2$:   CMPB    R0,#'1          ;^DIGIT MEANS SUBSTITUTE PARAM
007470  103414                          BLO     1$              ;BR IF NOT PARAM
007472  120027  000060G                 CMPB    R0,#<60+MXCPRM> ;IN VALID RANGE?
007476  101011                          BHI     1$              ;BR IF NOT
                                        ;
                                        ;  Switch input to a parameter string
                                        ;
007500  042700  177760                  BIC     #177760,R0      ;LEAVE VALUE ONLY
007504  006300                          ASL     R0              ;CONVERT TO WORD TABLE INDEX
007506  016037  1777760 000000G         MOV     <PRMPNT-2>(R0),CURPRM ;SET CHAR STRING POINTER
007514  004737  000000G                 CALL    CHKABT          ;ALLOW ABORTS WHILE GETTING PARAMETERS
007520  000671                          BR      10$             ;GO GET 1ST CHAR FROM STRING
                                        ;
                                        ;  ^letter causes the letter to be converted to a control character
                                        ;
007522  120027  000101            1$:   CMPB    R0,#'A          ;IS THIS A LETTER?
007526  103414                          BLO     15$             ;BR IF NOT
007530  120027  000132                  CMPB    R0,#'Z
007534  101406                          BLOS    16$             ;BR IF UPPER-CASE LETTER
007536  120027  000141                  CMPB    R0,#141         ;SEE IF THIS IS A LOWER-CASE LETTER
007542  103406                          BLO     15$             ;BR IF NOT
007544  120027  000172                  CMPB    R0,#172
007550  101003                          BHI     15$             ;BR IF NOT LETTER
007552  042700  177740            16$:  BIC     #177740,R0      ;CONVERT ALPHA TO CONTROL CHAR
007556  000404                          BR      12$             ;RETURN CONTROL CHARACTER AS CHARACTER GOTTEN
                                        ;
                                        ;  ^ was followed by something we don't recognize.
                                        ;  Just return the ^ as the character gotten.
```

```
    115                                         ;
    116 007560  110037  000000G                 15$:    MOVB    R0,CFHOLD       ;"PUSH" THE NEXT CHARACTER
    117 007564  112700  000136                          MOVB    #'^,R0          ;RETURN "^" AS THIS CHARACTER
    118                                         ;
    119                                         ;  Finished
    120                                         ;
    121 007570  000241                          12$:    CLC                     ;INDICATE SUCCESSFUL RETURN
    122 007572  000401                                  BR      13$
    123 007574  000261                          14$:    SEC                     ;INDICATE UNSUCCESSFUL RETURN
    124 007576  000207                          13$:    RETURN
```

```
                                        .SBTTL  CFCHAR -- Do command file I/O
                                ;-----------------------------------------------------------------------
                                ;  CFCHAR is the lowest level routine called to get a character from
                                ;  a command file.  It does the actual I/O to read the command file
                                ;  and returns the next character from the buffer or from a parameter
                                ;  string if string substitution is being done.
                                ;
                                ;  Inputs:
                                ;   R1 = Current job index number.
                                ;
                                ;  Outputs:
                                ;   R0 = Next character from command file.
                                ;   C-flag set on return if end of file hit.
                                ;
007600  010246                  CFCHAR: MOV     R2,-(SP)
007602  013702  0000000G                MOV     CURPRM,R2       ;INPUT FROM PARAM STRING?
007606  001407                        BEQ     5$              ;BR IF NOT
                                ;
                                ;  INPUT IS COMING FROM A PARAMETER STRING
                                ;
007610  112200                        MOVB    (R2)+,R0        ;GET NEXT CHAR FROM STRING
007612  001403                        BEQ     6$              ;BR IF HIT END OF STRING
007614  010237  0000000G              MOV     R2,CURPRM       ;UPDATE CHAR POINTER
007620  000452                        BR      9$              ;RETURN
                                ;  HIT END OF PARAMETER STRING.
                                ;  SWITCH INPUT BACK TO CONTROL FILE.
007622  005037  0000000G        6$:     CLR     CURPRM          ;CLEAR PARAM STRING POINTER
                                ;
                                ;  GET CHARACTER FROM CONTROL FILE
                                ;
007626  013702  0000000G        5$:     MOV     CFPNT,R2        ;GET POINTER INTO BUFFER
007632  020227  0000000G        4$:     CMP     R2,#CFEND       ;HIT END OF BUFFER?
007636  103437                        BLO     1$              ;BRANCH IF NOT
                                ;  REACHED END OF BUFFER -- READ IN NEXT BLOCK.
007640  113746  000052                MOVB    @#52,-(SP)      ;SAVE I/O ERROR CODE
007644  005237  0000000G              INC     CFBLK           ;INC FILE BLOCK NUMBER
007650                                .READW  #CFARG,#CFCHAN,#CFBUF,#256.,CFBLK
007710  112637  000052                MOVB    (SP)+,@#52      ;REPLACE ERROR CELL
007714  103006                        BCC     3$              ;BR IF NOT AT END OF FILE
                                ;
                                ;  End of file has been hit.
                                ;  Try to pop up to higher level file.
                                ;
007716  004737  010070'              CALL    CFPOP           ;TRY TO POP UP TO HIGHER LEVEL FILE
007722  005737  0000000G              TST     CFPNT           ;WAS THERE A HIGHER LEVEL FILE?
007726  001337                        BNE     5$              ;BR IF YES
007730  000410                        BR      11$
                                ;  GET NEXT CHAR FROM BUFFER
007732  012702  0000000G        3$:     MOV     #CFBUF,R2       ;RESET BUFFER POINTER
007736  112200                  1$:     MOVB    (R2)+,R0        ;GET NEXT CHAR
007740  001734                        BEQ     4$              ;IGNORE NULLS
007742  010237  0000000G        8$:     MOV     R2,CFPNT        ;SAVE NEW CHAR POINTER
                                ;  Successful return.
007746  000241                  9$:     CLC
007750  000401                        BR      12$
                                ;  Unsuccessful return.
007752  000261                  11$:    SEC                     ;SIGNAL UNSUCCESSFUL RETURN
```

```
     58 007754  012602                    12$:    MOV     (SP)+,R2
     59 007756  000207                            RETURN
```

```
                                        .SBTTL  CFTEST -- Determine if TT input is from file
                                ;-----------------------------------------------------------
                                ;  CFTEST IS CALLED TO DETERMINE IF TT INPUT IS COMING FROM
                                ;  A COMMAND FILE.
                                ;  IF YES, THE C-FLAG IS RESET ON RETURN.
                                ;  IF NO, THE C-FLAG IS SET ON RETURN.
                                ;  WHEN CALLED, R1 MUST CONTAIN THE USER INDEX NUMBER.
                                ;  ALL REGISTERS ARE PRESERVED.
                                ;
007760  005737  0000000       CFTEST: TST     CFPNT           ;INPUT FROM COMMAND FILE?
007764  001412                        BEQ     CFTNO           ;BR IF NOT
007766  032761  0000000 0000000 CFTST1: BIT     #$GTLIN,LSW4(R1);IS THIS A .GTLIN INPUT EMT?
007774  001004                        BNE     2$              ;BR IF YES
007776  032761  0000000 0000000       BIT     #$CFALL,LSW4(R1);GET .TTYIN INPUT FROM @FILE?
010004  001402                        BEQ     CFTNO           ;BR IF NOT
010006  000241                2$:     CLC                     ;SAY INPUT COMING FROM FILE
010010  000207                        RETURN
010012  000261                CFTNO:  SEC                     ;SAY INPUT NOT COMING FROM FILE
010014  000207                        RETURN
```

```
 1                                              .SBTTL  CFSTOP -- Suspend command file input
 2                                      ;-----------------------------------------------------------------
 3                                      ;  Suspend command file input by setting CFPNT=0.
 4                                      ;  All registers are preserved.
 5                                      ;
 6 010016  010046                       CFSTOP: MOV     R0,-(SP)
 7                                      ;
 8                                      ;  Say input not coming from a command file
 9                                      ;
10 010020  005037  000000G                     CLR     CFPNT           ;Suspend command file input
11                                      ;
12                                      ;  Clear command-file-active status flag in RMON cell
13                                      ;
14 010024  013700  000000G                     MOV     CXTRMN,R0       ;Get virtual address of RMON in cxt blk
15 010030  042760  000000G 000000G            BIC     #CFACFL,R$CFST(R0) ;Say command file not active
16                                      ;
17                                      ;  Finished
18                                      ;
19 010036  012600                              MOV     (SP)+,R0
20 010040  000207                              RETURN
21
22                                              .SBTTL  CFSTRT -- Restart command file input
23                                      ;-----------------------------------------------------------------
24                                      ;  Restart command file input by storing a non-zero value into CFPNT.
25                                      ;
26                                      ;  Inputs:
27                                      ;   R0 = Value to store into CFPNT
28                                      ;
29 010042  010046                       CFSTRT: MOV     R0,-(SP)
30                                      ;
31                                      ;  Restart command file input
32                                      ;
33 010044  010037  000000G                     MOV     R0,CFPNT        ;Set command file buffer pointer
34 010050  001405                              BEQ     9$              ;Br if not starting command file
35                                      ;
36                                      ;  Set command-file-active flag in RMON status cell
37                                      ;
38 010052  013700  000000G                     MOV     CXTRMN,R0       ;Get virtual address of RMON in cxt blk
39 010056  052760  000000G 000000G            BIS     #CFACFL,R$CFST(R0) ;Set command-file-active flags
40                                      ;
41                                      ;  Finished
42                                      ;
43 010064  012600                       9$:    MOV     (SP)+,R0
44 010066  000207                              RETURN
```

```
                                        .SBTTL  CFPOP  -- Pop up to next command file
                                ;-------------------------------------------------------------
                                ;  CFPOP is called to close the current command file and pop up to
                                ;  the next higher command file.
                                ;
010070  010146                  CFPOP:  MOV     R1,-(SP)
010072  010346                          MOV     R3,-(SP)
010074  010446                          MOV     R4,-(SP)
010076  010546                          MOV     R5,-(SP)
010100  113746  000052                  MOVB    @#52,-(SP)      ;SAVE I/O ERROR CODE CELL
010104  113701  0000000                 MOVB    CORUSR,R1       ;GET JOB INDEX NUMBER
010110  042761  0000000 0000000         BIC     #$CFCCL,LSW4(R1);SAY WE ARE NOT GETTING CHARS FROM CCL COMMAND
010116  005737  0000000                 TST     CFPNT           ;IS A COMMAND FILE IN USE NOW?
010122  001002                          BNE     11$             ;Br if yes
010124  000137  010676'                JMP     9$
010130  013705  0000000         11$:    MOV     CXTRMN,R5       ;GET ADDRESS OF SIMULATED RMON DATA
010134  113765  0000000 0000000         MOVB    CFIND,R$INST(R5);RESTORE IND STATUS BYTE
010142  105037  0000000                 CLRB    CFHOLD          ;CLEAR ANY COMMAND FILE HOLDING CHAR
                                ;
                                ;  Close currently open file
                                ;
010146  032761  0000000 0000000         BIT     #$CFOPN,LSW4(R1);IS THE COMMAND FILE CHANNEL OPEN?
010154  001406                          BEQ     1$              ;BR IF NOT
010156                                  .PURGE  #CFCHAN         ;CLOSE CURRENT FILE
010164  042761  0000000 0000000         BIC     #$CFOPN,LSW4(R1);SAY COMMAND FILE CHANNEL IS NOW CLOSED
                                ;
                                ;  See if there is a higher level command file to restore
                                ;
010172  105737  0000000         1$:     TSTB    CFNEST          ;ANY HIGHER LEVEL COMMAND FILES?
010176  001060                          BNE     2$              ;BR IF YES
                                ;
                                ;  There are no higher level command files
                                ;
010200  004737  010016'                 CALL    CFSTOP          ;Say no data coming from command file
010204  042761  000000C 0000000         BIC     #<$CFALL!$CFSOT>,LSW4(R1) ;CLEAR COMMAND FILE CONTROL FLAGS
010212  032761  0000000 0000000         BIT     #$LOFCF,LSW9(R1);Are we leaving a log off command file?
010220  001006                          BNE     15$             ;If so, do not restore terminal input
010222  042761  0000000 0000000         BIC     #$NOIN,LSW3(R1) ;ALLOW INPUT TO BE ACCEPTED FOR LINE
010230  042761  0000000 0000000         BIC     #$SUCF,LSW9(R1) ;Say we are no longer executing startup file
010236  005000                  15$:    CLR     R0              ;Init privilege vector index
010240  016060  0000000 0000000 10$:    MOV     PRIVS0(R0),PRIVF0(R0);Reset command file priv to set priv
010246  032761  0000000 0000000         BIT     #$INKMN,LSW4(R1);Are we in TSKMON now?
010254  001403                          BEQ     12$             ;Br if not
010256  016060  0000000 0000000         MOV     PRIVS0(R0),PRIVC0(R0) ;Reset current privileges
010264  062700  000002          12$:    ADD     #2,R0           ;Increment word index
010270  020027  000000C                 CMP     R0,#PVNPW*2     ;Done all?
010274  103761                          BLO     10$             ;Loop if more
010276  005037  0000000                 CLR     AFCF            ;Clear command file attribute flags
010302  032761  0000000 0000000         BIT     #$INKMN,LSW4(R1);Are we in TSKMON now?
010310  001572                          BEQ     9$              ;Br if not
010312  042761  0000000 0000000         BIC     #$SCCA,LSW5(R1) ;Clear abort-suppression flag
010320  042761  0000000 0000000         BIC     #$NOWIN,LSW11(R1);Clear window suppression
010326  105737  0000000                 TSTB    SUCF2           ;Is there a pending secondary file?
010332  001561                          BEQ     9$              ;Br if not
010334  004737  0000000                 CALL    STOP            ;Reenter KMON to start secondary file
                                ;
                                ;  Reopen next higher level file
```

```
                                        ;  Restore parameter pointers
                                        ;
010340  013705  000000G         2$:     MOV     CFSP,R5         ;GET COMMAND FILE STACK POINTER
010344  012703  000000G                 MOV     #PRMPNT,R3      ;POINT TO PARAM POINTER CELLS
010350  012504                  4$:     MOV     (R5)+,R4        ;GET A PARAMETER POINTER
010352  001402                          BEQ     3$              ;BR IF END OF LIST HIT
010354  010423                          MOV     R4,(R3)+        ;RESTORE POINTER
010356  000774                          BR      4$              ;GO BACK AND DO NEXT ONE
010360  020327  000000G         3$:     CMP     R3,#LSTPRM      ;ZERO ALL OTHER PARAMETER POINTERS
010364  103002                          BHIS    5$
010366  005023                          CLR     (R3)+
010370  000773                          BR      3$
                                        ;
                                        ;  Restore parameter strings
                                        ;
010372  012504                  5$:     MOV     (R5)+,R4        ;GET ADDRESS OF END OF STRING
010374  010437  000000G                 MOV     R4,PBFEND
010400  020427  000000G         7$:     CMP     R4,#PRMBUF      ;RESTORED ALL OF STRING?
010404  101402                          BLOS    6$              ;BR IF YES
010406  012544                          MOV     (R5)+,-(R4)     ;POP STRING OFF OF STACK
010410  000773                          BR      7$
010412  012537  000000G         6$:     MOV     (R5)+,CURPRM    ;POP POINTER INTO STRING
                                        ;
                                        ;  Restore IND status flags
                                        ;
010416  012537  000000G                 MOV     (R5)+,CFIND     ;RESTORE IND STATUS FLAGS
                                        ;
                                        ;  Reset command file control flags
                                        ;
010422  012704  000000G                 MOV     #CFLFL4,R4      ;GET CONTROL FLAG MASK
010426  040461  000000G                 BIC     R4,LSW4(R1)     ;CLEAR THOSE FLAGS
010432  005104                          COM     R4              ;MASK ALL BUT THOSE FLAGS
010434  040415                          BIC     R4,(R5)
010436  052561  000000G                 BIS     (R5)+,LSW4(R1)  ;SET DESIRED FLAGS
                                        ;
                                        ;  Restore command file attribute flags
                                        ;
010442  012537  000000G                 MOV     (R5)+,AFCF      ;Restore command file attribute flags
010446  052761  000000G 000000G         BIS     #$SCCA,LSW5(R1) ;Assume ctrl-C abort suppression wanted
010454  032737  000000G 000000G         BIT     #AF$CCA,AFCF    ;Is abort suppression wanted?
010462  001003                          BNE     13$             ;Br if yes
010464  042761  000000G 000000G         BIC     #$SCCA,LSW5(R1) ;Clear abort-suppression flag
010472  032737  000000G 000000G 13$:     BIT     #AF$NPW,AFCF    ;Suppressing process windowing?
010500  001003                          BNE     14$             ;Br if yes
010502  042761  000000G 000000G         BIC     #$NOWIN,LSW11(R1);Clear window suspend flag
                                        ;
                                        ;  Restore command file privileges
                                        ;
010510  005000                  14$:    CLR     R0              ;Init vector index
010512  011560  000000G         8$:     MOV     (R5),PRIVF0(R0) ;Restore each privilege word
010516  012560  000000G                 MOV     (R5)+,PRIVC0(R0);Reset current privilege too
010522  062700  000002                  ADD     #2,R0           ;Increment index
010526  020027  00000C                  CMP     R0,#PVNPW*2     ;Done all?
010532  103767                          BLO     8$              ;Loop if more
                                        ;
                                        ;  Restore buffer pointer information
                                        ;
```

```
    115 010534  012500                          MOV     (R5)+,R0        ;POINTER INTO BUFFER
    116 010536  004737  010042'                 CALL    CFSTRT          ;Reset command file pointer
    117 010542  012537  000000G                 MOV     (R5)+,CFBLK     ;CURRENT BLOCK NUMBER
    118                                 ;
    119                                 ;  Reopen the command file
    120                                 ;
    121 010546                                  .REOPEN #CFARG,#CFCHAN,R5 ;REOPEN COMMAND FILE
    122 010564  062705  000012                  ADD     #10.,R5         ;POP SAVE STATUS INFO
    123 010570  052761  000000G 000000G         BIS     #$CFOPN,LSW4(R1);SAY COMMAND FILE CHANNEL IS OPEN
    124                                 ;
    125                                 ;  Reread current buffer from file
    126                                 ;
    127 010576                                  .READW  #CFARG,#CFCHAN,#CFBUF,#256.,CFBLK
    128                                 ;
    129                                 ;  Finished restoring file
    130                                 ;
    131 010636  010537  000000G                 MOV     R5,CFSP         ;SAVE UPDATED STACK POINTER
    132 010642  105337  000000G                 DECB    CFNEST          ;SAY ONE LESS FILE ON STACK
    133                                 ;
    134                                 ;  If we just reached the end of a command file created to hold
    135                                 ;  an expanded IND command and KMON is trying to read another
    136                                 ;  command, call STOP to force KMON to go back to IND to get the
    137                                 ;  next command before continuing on with an outer level
    138                                 ;  command file.
    139                                 ;
    140 010646  032761  000000G 000000G         BIT     #$INKMN,LSW4(R1);IS KMON RUNNING?
    141 010654  001410                          BEQ     9$              ;BR IF NOT
    142 010656  013705  000000G                 MOV     CXTRMN,R5       ;GET ADDRESS OF SIMULATED RMON DATA
    143 010662  132765  000000G 000000G         BITB    #IN$ACT,R$INST(R5);IS IND ACTIVE?
    144 010670  001402                          BEQ     9$              ;BR IF NOT
    145 010672  004737  000000G                 CALL    STOP            ;REENTER KMON TO FORCE REENTRY OF IND
    146 010676  112637  000052          9$:     MOVB    (SP)+,@#52      ;RESTORE I/O ERROR CELL
    147 010702  012605                          MOV     (SP)+,R5
    148 010704  012604                          MOV     (SP)+,R4
    149 010706  012603                          MOV     (SP)+,R3
    150 010710  012601                          MOV     (SP)+,R1
    151 010712  000207                          RETURN
```

```
                                        .SBTTL  LOGCHR -- Write character to log file
                                ;-----------------------------------------------------------------------
                                ;  LOGCHR is called to write a character to the log file.
                                ;  Control characters are converted into ^character sequences.
                                ;
                                ;  Inputs:
                                ;   R0 = Character to be written to log.
                                ;
 010714  010046                 LOGCHR: MOV     R0,-(SP)
                                ;
                                ;  See if this is a control character
                                ;
 010716  042700  177400                 BIC     #^C377,R0       ;Strip character down to 8 bits
 010722  120027  000003                 CMPB    R0,#3           ;Is character control-C?
 010726  001403                         BEQ     1$              ;Br if yes
 010730  120027  000032                 CMPB    R0,#32          ;Is character control-Z?
 010734  001007                         BNE     2$              ;Br if not
 010736  112700  000136         1$:     MOVB    #'^,R0          ;Log "^"
 010742  004737  011012'                CALL    LOGCH1
 010746  011600                         MOV     (SP),R0         ;Get back original character
 010750  062700  000100                 ADD     #100,R0         ;Convert to printing character
 010754  004737  011012'         2$:     CALL    LOGCH1          ;Log the character
                                ;
                                ;  Finished
                                ;
 010760  012600                         MOV     (SP)+,R0
 010762  000207                         RETURN

                                ;-----------------------------------------------------------------------
                                ;  LOGCR is called to send a carriage-return and line-feed sequence
                                ;  to the log file.
                                ;
 010764  010046                 LOGCR:  MOV     R0,-(SP)
 010766  112700  000000G                MOVB    #CR,R0          ;Log carriage-return
 010772  004737  011012'                CALL    LOGCH1
 010776  112700  000000G                MOVB    #LF,R0          ;Log line-feed
 011002  004737  011012'                CALL    LOGCH1
 011006  012600                         MOV     (SP)+,R0
 011010  000207                         RETURN
```

```
 1                                      ;-----------------------------------------------------------------------
 2                                      ;  LOGCH1 is called to move a character to the log buffer.
 3                                      ;  No tests or conversions are performed on the character.
 4                                      ;  No logging is done if echo suppression is in effect.
 5                                      ;
 6                                      ;  Inputs:
 7                                      ;   R0 = Character to be written.
 8                                      ;
 9 011012  010246                       LOGCH1: MOV     R2,-(SP)
10                                      ;
11                                      ;  Check to see if log file output has been suspended (NOWRITE option)
12                                      ;
13 011014  032737  0000000 0000000             BIT     #LF$WRT,LOGFLG  ;Has log file been suspended?
14 011022  001453                              BEQ     9$              ;Br if yes
15                                      ;
16                                      ;  Get current buffer pointer and make sure buffer is not full
17                                      ;
18 011024  013702  0000000             1$:     MOV     LOGPTR,R2       ;Get current log file buffer pointer
19 011030  001450                              BEQ     9$              ;Br if not doing logging
20 011032  020227  0000000                     CMP     R2,#LOGEND      ;Is buffer full?
21 011036  103442                              BLO     1$              ;Br if not
22                                      ;
23                                      ;  Log file buffer is full.  Write it to the log file.
24                                      ;
25 011040  012702  0000000                     MOV     #LOGBUF,R2      ;Point to front of buffer
26 011044  010046                              MOV     R0,-(SP)
27 011046  113746  000052                      MOVB    @#52,-(SP)      ;Save job's error cell
28 011052                                      .WRITW  #CFARG,#LOGCHN,R2,#256.,LOGBLK ;Write block to log file
29 011110  103010                              BCC     2$              ;Br if no write error
30 011112  012705  177753                      MOV     #-25,R5         ;Abort job if error writing to log file
31 011116  013704  0000000                     MOV     EMTADR,R4       ;Get address of EMT
32 011122  005037  0000000                     CLR     LOGPTR          ;Say we have stopped using log file
33 011126  000137  0000000                     JMP     ABORT           ;Abort the job
34 011132  005237  0000000            2$:     INC     LOGBLK          ;Advance log file block number
35 011136  112637  000052                      MOVB    (SP)+,@#52      ;Restore error cell
36 011142  012600                              MOV     (SP)+,R0
37                                      ;
38                                      ;  Move character to log buffer
39                                      ;
40 011144  110022                      1$:     MOVB    R0,(R2)+        ;Move char to log file buffer
41 011146  010237  0000000                     MOV     R2,LOGPTR       ;Save new buffer pointer
42                                      ;
43                                      ;  Finished
44                                      ;
45 011152  012602                      9$:     MOV     (SP)+,R2
46 011154  000207                              RETURN
```

```
                                        .SBTTL  ILWAIT -- Wait for activation char from terminal
                                ;-----------------------------------------------
                                ;  ILWAIT WAITS UNTIL AN ACTIVATION CHARACTER IS RECEIVED
                                ;  FOR CURRENT USER.  ALL REGISTERS ARE PRESERVED.
                                ;  WHEN CALLED R1 MUST CONTAIN THE USER INDEX #.
                                ;
011156  032761  000000G 000000G ILWAIT: BIT     #$DETCH,LSW(R1) ;IS THIS A DETACHED JOB?
011164  001405                         BEQ     15$             ;BR IF NOT
011166  052761  000000G 000000G        BIS     #$DISCN,LSW(R1) ;FORCE JOB LOGOFF
011174  004737  000000G               CALL    STOP            ;HALT DETACHED JOBS THAT WANT TERMINAL INPUT
                                ;
                                ;  If we previously stopped input from silo buffer, Restart it now
                                ;  if we are about to run out of characters.
                                ;
011200  032761  000000G 000000G 15$:    BIT     #$XSTOP,LSW6(R1);DID WE SUSPEND TRANSMISSION TO US?
011206  001410                         BEQ     11$             ;BR IF NOT
011210  042761  000000G 000000G        BIC     #$XSTOP,LSW6(R1);RESTART INPUT
011216  052761  000000G 000000G        BIS     #$NDICP,LSW10(R1);Say line needs input character servicing
011224  005237  000000G               INC     NEDCDI          ;Say input processing needed
                                ;
                                ;  If we are in deferred echo mode, echo any pending
                                ;  Characters on last line.
                                ;
011230  004737  011346'         11$:    CALL    DFRREL          ;Release deferred echo mode
                                ;
                                ;  Suspend user till activation character received.
                                ;
011234  004737  017466'                 CALL    SIGWAT          ;SIGNAL VIRTUAL LINE WAIT CONDITION
011240  042761  000000G 000000G        BIC     #$NOIN,LSW3(R1) ;ALLOW INPUT TO BE ACCEPTED FOR LINE
011246  042761  000000G 000000G        BIC     #$SUCF,LSW9(R1) ;Say we are no longer executing startup file
011254  004737  000000G       13$:    CALL    CHKABT          ;SEE IF JOB HAS BEEN ABORTED
011260                                 DISABL                  ;** DISABLE **
011266  005761  000000G               TST     LACTIV(R1)      ;GOT ANY ACTIVATION CHARS YET?
011272  001021                         BNE     1$              ;BR IF YES
011274  032761  000000G 000000G        BIT     #$NOINT,LSW7(R1);Does user want non-interactive execution?
011302  001006                         BNE     5$              ;Br if yes
011304  013761  000000G 000000G        MOV     VINTIO,LHIPCT(R1);RESET INTERACTIVE I/O LIMIT FOR JOB
011312  013761  000000G 000000G        MOV     VQUAN1,LITIME(R1);RESET INTERACTIVE CPU TIME LIMIT
011320  010046                 5$:     MOV     R0,-(SP)
011322  012700  000000G               MOV     #S$INWT,R0      ;GET WAITING-FOR-INPUT STATE
011326  004737  000000G               CALL    QHDSPN          ;SUSPEND JOB AND WAIT FOR ACTIVATION *ENABLE*
011332  012600                         MOV     (SP)+,R0
011334  000747                         BR      13$             ;NOW CHECK AGAIN
011336                         1$:     ENABL                   ;** ENABLE **
011344  000207                         RETURN
```

```
                                        .SBTTL  DFRREL -- Release deferred echo mode
                                ;-----------------------------------------------------------------------
                                ;  DFRREL is called to release deferred echo mode and to echo any
                                ;  pending deferred characters.
                                ;
                                ;  Inputs:
                                ;   R1 = Job index number
                                ;
011346  010246                  DFRREL: MOV     R2,-(SP)
011350  010346                          MOV     R3,-(SP)
011352  010446                          MOV     R4,-(SP)
                                ;
                                ;  See if we are currently doing deferred echoing
                                ;
011354                                  DISABL                  ;** DISABLE INTERRUPTS **
011362  032761  0000000 0000000         BIT     #$DODFR,LSW3(R1);ARE WE IN DEFERRED ECHO MODE?
011370  001460                          BEQ     6$              ;BRANCH IF NOT
011372  005761  0000000                 TST     LACTIV(R1)      ;ARE THERE ANY PENDING ACTIVATION CHARS?
011376  001055                          BNE     6$              ;BR IF YES -- DON'T ECHO IF PENDING ACTIV
011400                                  ENABL                   ;** ENABLE INTERRUPTS **
                                ;
                                ;  We are in deferred echo mode
                                ;
011406  042761  0000000 0000000         BIC     #$GCECO,LSW3(R1);RESET GETCHR ECHOING
011414  052761  0000000 0000000         BIS     #$1STCH,LSW3(R1);SAY WE'VE GOT 1ST CHAR
011422  116161  0000000 0000000         MOVB    LCOL(R1),LINCUR(R1);SAVE CURSOR POSITION
011430  005004                          CLR     R4              ;R4 IS FLAG FOR ESC SEQ CHARS
011432  005003                          CLR     R3
011434  016102  0000000                 MOV     LINPNT(R1),R2   ;POINT TO START OF PENDING CHARS
                                ;
                                ;  Begin loop to echo all pending characters
                                ;
011440  020361  0000000         5$:     CMP     R3,LINCNT(R1)   ;ECHOED ALL PENDING?
011444  103027                          BHIS    4$              ;BRANCH IF FINISHED
011446  004737  016312'                 CALL    FETCHR          ;GET NEXT CHAR FROM TT INPUT BUFFER
011452  005704                          TST     R4              ;IS THIS CHAR PART OF ESC SEQ?
011454  001015                          BNE     8$              ;BR IF YES
011456  120027  0000000                 CMPB    R0,#ESCFLG      ;FLAG THAT NEXT CHAR PART OF ESC SEQUENCE?
011462  001006                          BNE     9$              ;BR IF NOT
011464  032761  0000000 0000000         BIT     #$VTESC,LSW5(R1);Are we activating on escape sequences?
011472  001402                          BEQ     9$              ;Br if not -- Treat ESCFLG like normal char
011474  005204                          INC     R4              ;SET ESC SEQ FLAG
011476  000410                          BR      10$
011500  004737  017414'         9$:     CALL    CVTLC           ;CONVERT LOWER CASE TO UPPER CASE
011504  004737  007030'                 CALL    GCECHO          ;ECHO IT
011510  005004                  8$:     CLR     R4              ;CLEAR ESC SEQ FLAG
011512  032700  0000000                 BIT     #ACFLAG,R0      ;IS THIS AN ACTIVATION CHAR?
011516  001005                          BNE     6$              ;BRANCH IF YES
011520  005203                  10$:    INC     R3              ;COUNT CHARS ECHOED
011522  000746                          BR      5$
                                ;
                                ;  Release deferred echo mode
                                ;
011524  042761  0000000 0000000 4$:     BIC     #$DODFR,LSW3(R1);BEGIN IMMEDIATE CHAR ECHOING
                                ;
                                ;  Finished
                                ;
```

```
     58 011532                          6$:     ENABL                   ;** ENABLE INTERRUPTS **
     59 011540  012604                          MOV     (SP)+,R4
     60 011542  012603                          MOV     (SP)+,R3
     61 011544  012602                          MOV     (SP)+,R2
     62 011546  000207                          RETURN
```

```
                                        .SBTTL
                                        .SBTTL  **  Fork Level Input Character Processing **
                                        .SBTTL  TTINCP -- Process received input characters
                                ;-----------------------------------------------------------------
                                ;  TTINCP is called at fork level after each received character
                                ;  has been stored in the high speed input ring buffer.
                                ;  The function of TTINCP is to remove characters from the
                                ;  high speed input ring buffer and perform the TSX-Plus
                                ;  character processing which will eventually cause the character
                                ;  to be stored in the input ring buffer for the line.
                                ;
                                ;  Inputs:
                                ;   R4 = Line index number
                                ;
011550  010146                  TTINCP: MOV     R1,-(SP)
011552  010546                          MOV     R5,-(SP)
011554  010401                          MOV     R4,R1           ;Carry line index number in R1 too
                                ;
                                ;  See if TT input ring buffer is full
                                ;
011556  032764  0000006 0000006 5$:     BIT     #$XSTOP,LSW6(R4);Has input been suspended due to buffer full
011564  001007                          BNE     9$              ;Br if yes
                                ;
                                ;  Get next character from input silo
                                ;
011566  004777  0000006                 CALL    @SILFET         ;Get next character from input silo
011572  103404                          BCS     9$              ;Br if no characters in silo
011574  010005                          MOV     R0,R5           ;Put character in R5 for GOTCHR
                                ;
                                ;  Now call main routine to process a character for this line
                                ;
011576  004737  011612'                 CALL    GOTCHR          ;Process a character for this line
                                ;
                                ;  Finished processing a character.
                                ;  Go back and see if there are any more characters to process.
                                ;
011602  000765                          BR      5$              ;Check for more characters to process
                                ;
                                ;  Finished processing all pending characters for this line.
                                ;
                                ;
011604  012605                  9$:     MOV     (SP)+,R5
011606  012601                          MOV     (SP)+,R1
011610  000207                          RETURN
```

```
                                ;-----------------------------------------------------------------------
                                ;  A character has been received from a line.
                                ;  Check to see if the line is active or needs to be started.
                                ;
                                ;  Inputs:
                                ;   R4 = Physical line index number.
                                ;   R5 = Received character.
                                ;
011612  010146                  GOTCHR: MOV     R1,-(SP)
                                ;
                                ;  Ignore the character if system initialization is not yet complete
                                ;
011614  105737  000000G                 TSTB    INITFLG         ;Is system initialization finished yet?
011620  001052                          BNE     9$              ;Br if not
                                ;
                                ;  See if this line has been initialized yet
                                ;
011622  116401  000000G                 MOVB    LNMAP(R4),R1    ;Get virtual line index number
011626  032761  000000G 000000G         BIT     #$DILUP,LSW(R1) ;Has line been started yet?
011634  001033                          BNE     1$              ;Br if yes
                                ;
                                ;  Line has not been initialized yet.
                                ;  See if we should start it now.
                                ;
011636  105737  000000G                 TSTB    STPFLG          ;Is a system shutdown in progress?
011642  001041                          BNE     9$              ;Br if yes -- Don't start any lines
011644  032764  000000G 000000G         BIT     #$DEAD,LSW3(R4) ;Is this line marked as dead?
011652  001035                          BNE     9$              ;Br if yes
                                ;
                                ;  See if we should do autobaud speed selection for this line
                                ;
011654  032764  000000G 000000G         BIT     #$AUTO,ILSW2(R4);Is autobaud speed selection wanted?
011662  001405                          BEQ     3$              ;Br if not
011664                                  OCALL   AUTSPD          ;Do autobaud speed selection
011672  103007                          BCC     2$              ;Br if we should start the line now
011674  000424                          BR      9$              ;Do not start the line yet
011676  120527  000000G         3$:     CMPB    R5,#CR          ;Is character Carriage-return?
011702  001403                          BEQ     2$              ;Br if yes
011704  120527  000000G                 CMPB    R5,#CTRLC       ;Is character ctrl-C?
011710  001016                          BNE     9$              ;Br if not
                                ;
                                ;  Start up a previously inactive line
                                ;
011712  005000                  2$:     CLR     R0              ;No secondary start-up command file
011714                                  OCALL   INITLN          ;Initialize the line
011722  000403                          BR      5$
                                ;
                                ;  See if we are ignoring trash characters that may be received after
                                ;  the autobaud start-up character.
                                ;
011724  005764  000000G         1$:     TST     LABTIM(R4)      ;Is autobaud masking trash characters?
011730  001404                          BEQ     4$              ;Br if not
011732  042761  000000G 000000G 5$:     BIC     #$RBRK,LSW10(R1);Clear break-received flag
011740  000402                          BR      9$              ;Wait for autobaud mask time to end
                                ;
                                ;  Process an input character for an active line
                                ;
```

```
     58 011742  004737  011752'          4$:     CALL    PRCHAR          ;Process the character
     59                                   ;
     60                                   ;  Finished
     61                                   ;
     62 011746  012601                    9$:     MOV     (SP)+,R1
     63 011750  000207                            RETURN
```

```
                                ;-----------------------------------------------------------------------
                                ;  Process a character received by an active line.
                                ;
                                ;  Inputs:
                                ;   R1 = Job's virtual line index number.
                                ;   R5 = Received character.
                                ;
011752  010246                  PRCHAR: MOV     R2,-(SP)
011754  010346                          MOV     R3,-(SP)
011756  010446                          MOV     R4,-(SP)
                                ;
                                ;  See if we received a real break (long space).
                                ;
011760  016104  000000G                 MOV     LNPRIM(R1),R4   ;Get primary line #
011764  032764  000000G 000000G         BIT     #$RBRK,LSW10(R4);Did we receive a break?
011772  001406                          BEQ     1$              ;Br if not
011774  004737  017544'                 CALL    SIGBRK          ;Signal that we received a break
012000  042764  000000G 000000G         BIC     #$RBRK,LSW10(R4);Clear break-received flag
012006  000551                          BR      PRCEND          ;Finished with break character
                                ;
                                ;  Mask character to 8 bits and ignore nulls
                                ;
012010  042705  177400          1$:     BIC     #^C377,R5       ;Mask character to 8 bits
012014  001546                          BEQ     PRCEND          ;Ignore null characters
                                ;
                                ;  If debugger is in control, bypass some special character processing
                                ;
012016  032761  000000G 000000G         BIT     #$DBGMD,LSW6(R1);Is debugging program in control?
012024  001051                          BNE     CKCW            ;If yes then bypass some checking
                                ;
                                ;  See if user defined an asynchronous break character
                                ;
012026  016102  000000G                 MOV     LBRKCH(R1),R2   ;Did user define an asynch break char?
012032  001405                          BEQ     CKSPAC          ;Br if not
012034  020502                          CMP     R5,R2           ;Is this the break character?
012036  001003                          BNE     CKSPAC          ;Br if not
012040  004737  017544'                 CALL    SIGBRK          ;Tell user that break char was received
012044  000532                          BR      PRCEND
                                ;
                                ;  See if this is a user-defined activation character
                                ;
012046  016102  000000G         CKSPAC: MOV     LNSPAC(R1),R2   ;Get number of user-defined activation chars
012052  001424                          BEQ     CKVTES          ;Br if there are none
012054  010500                          MOV     R5,R0           ;Get current character
012056  004737  017414'                 CALL    CVTLC           ;Convert to upper-case if needed
012062  016103  000000G                 MOV     LSPACT(R1),R3   ;Get pointer to table of activation chars
012066  120023                  1$:     CMPB    R0,(R3)+        ;Is this an activation character?
012070  001402                          BEQ     2$              ;Br if yes
012072  077203                          SOB     R2,1$           ;Loop if more to check
012074  000404                          BR      CKHIIN          ;This is not a user-defined activation char
012076  010005                  2$:     MOV     R0,R5           ;Get converted character to R5
012100  004737  016172'                 CALL    STRACT          ;Store the activation character
012104  000512                          BR      PRCEND
                                ;
                                ;  See if we are in high-efficienty mode
                                ;
012106  032761  000000G 000000G CKHIIN: BIT     #$HITTY,LSW4(R1);Are we in high-efficiency mode?
```

```
     58 012114  001403                          BEQ     CKVTES          ;Br if not
     59 012116  004737  015674'                 CALL    STRCHR          ;Store the character
     60 012122  000503                          BR      PRCEND
     61                                ;
     62                                ;  See if this is a VT52 escape-letter sequence
     63                                ;
     64 012124  032761  000000@ 000000@ CKVTES: BIT     #$VTESC,LSW5(R1);Activate on esc-letter sequence?
     65 012132  001406                          BEQ     CKCW            ;Br if not
     66 012134  004737  017214'                 CALL    SCACHK          ;Are we in single character activ mode?
     67 012140  103403                          BCS     CKCW            ;Br if yes
     68 012142  004737  015202'                 CALL    CKVTAC          ;Check for escape-letter sequence
     69 012146  103071                          BCC     PRCEND          ;Br if char was part of escape sequence
     70                                ;
     71                                ;  Check for request to switch to a virtual line
     72                                ;
     73 012150  120537  000000@       CKCW:    CMPB    R5,VVLSCH       ;Is this char a request to switch to vir line?
     74 012154  001024                          BNE     1$              ;Br if not
     75 012156  032761  000000@ 000000@       BIT     #$CTRLW,LSW3(R1);Was last character also control-W?
     76 012164  001414                          BEQ     2$              ;Br if not
     77 012166  042761  000000@ 000000@       BIC     #$1ESC,LSW(R1)  ;Say last char was not escape
     78 012174  042761  000000@ 000000@       BIC     #$CTRLW,LSW3(R1);Say last char not control-W
     79 012202  004737  017376'                 CALL    SCACHR          ;See if we are in single-char activation mode
     80 012206  103451                          BCS     PRCEND          ;Br if in single-char activation mode
     81 012210  004737  015656'                 CALL    INCHR           ;Pass control-W to program as normal char
     82 012214  000446                          BR      PRCEND
     83 012216  052761  000000@ 000000@ 2$:     BIS     #$CTRLW,LSW3(R1);Remember last char was control-W
     84 012224  000442                          BR      PRCEND
     85 012226  032761  000000@ 000000@ 1$:     BIT     #$CTRLW,LSW3(R1);Was control-W the last character?
     86 012234  001420                          BEQ     CKICTL          ;Br if not
     87 012236  020527  000037                  CMP     R5,#37          ;Is current character a control character?
     88 012242  101415                          BLOS    CKICTL          ;Br if yes
     89 012244  042761  000000@ 000000@       BIC     #$CTRLW,LSW3(R1);Say control-W is not the last char
     90 012252  162705  000060                  SUB     #'0,R5          ;Convert line # digit to binary value
     91 012256  002425                          BLT     PRCEND          ;Br if too small
     92 012260  020527  000000@                 CMP     R5,#MAXSEC      ;Don't exceed max line # allowed
     93 012264  003022                          BGT     PRCEND          ;Br if too large
     94 012266                                  OCALL   DOSWIT          ;Switch to a virtual line
     95 012274  000416                          BR      PRCEND
     96                                ;
     97                                ;  Determine if this is a normal or control character
     98                                ;
     99 012276  020527  000037       CKICTL:  CMP     R5,#37          ;Is this a normal or control char?
    100 012302  101003                          BHI     1$              ;Br if not control character
    101 012304  004737  012560'                 CALL    DOCTRL          ;Process a control character
    102 012310  000410                          BR      PRCEND
    103 012312  120527  000000@       1$:      CMPB    R5,#RUBOUT      ;Is this a rubout character?
    104 012316  001003                          BNE     2$              ;Br if not
    105 012320  004737  014644'                 CALL    ICPRUB          ;Process rubout character
    106 012324  000402                          BR      PRCEND
    107 012326  004737  012342'       2$:      CALL    REGCHR          ;Process a normal character
    108                               ;
    109                               ;  Finished
    110                               ;
    111 012332  012604                PRCEND:  MOV     (SP)+,R4
    112 012334  012603                         MOV     (SP)+,R3
    113 012336  012602                         MOV     (SP)+,R2
    114 012340  000207                         RETURN
```

```
                        .SBTTL  REGCHR -- Process normal characters
                ;-----------------------------------------------------------------------
                ;  Process normal (non-control) characters.
                ;
                ;  Inputs:
                ;   R1 = Virtual line index number.
                ;   R5 = Current input character.
                ;
012342  010246  REGCHR: MOV     R2,-(SP)
012344  010546          MOV     R5,-(SP)
012346  010500          MOV     R5,R0           ;Copy the character
                ;
                ;  Ignore all input while processing logoff command file
                ;
012350  032761  0000000 0000000         BIT     #$LOFCF,LSW9(R1);Are we processing a logoff command file?
012356  001075          BNE     9$              ;Br if yes -- ignore the character
                ;
                ;  Say last character was not control-C
                ;
012360  042761  0000000 0000000         BIC     #$1ESC,LSW(R1)  ;Say last char was not escape
012366  042761  0000000 0000000         BIC     #$1CTLC,LSW5(R1);Say last character was not control-C
                ;
                ;  See if we are in single character activation mode
                ;
012374  004737  017254'                 CALL    SLCHK           ;Are we in single-char activation mode or SL?
012400  103003                          BCC     6$              ;Br if not
012402  004737  016252'                 CALL    STRSNG          ;Store character and activate
012406  000461                          BR      9$
                ;
                ;  See if ODT is in control
                ;
012410  032761  0000000 0000000 6$:     BIT     #$ODTMD,LSW4(R1);Is ODT activation in effect?
012416  001413                          BEQ     3$              ;Br if not
012420  004737  015414'                 CALL    CHKODT          ;Is this an ODT activation char?
012424  103010                          BCC     3$              ;Br if not
012426  120027  000040                  CMPB    R0,#40          ;Is this a control character?
012432  103402                          BLO     4$              ;Br if yes -- Don't echo it
012434  004737  017032'                 CALL    ECHO            ;Echo the character
012440  004737  016252'          4$:    CALL    STRSNG          ;Store character and activate
012444  000442                          BR      9$
                ;
                ;  Check for activation on input of a certain number of characters
                ;
012446  016102  0000000         3$:     MOV     LAFSIZ(R1),R2   ;Was field width activation specified?
012452  001416                          BEQ     1$              ;Br if not
012454  005302                          DEC     R2              ;Is field full yet?
012456  020261  0000000                 CMP     R2,LINCNT(R1)
012462  101012                          BHI     1$              ;Br if not
012464  032761  0000000 0000000         BIT     #$DBGMD,LSW6(R1);Is a debugger running?
012472  001006                          BNE     1$              ;Br if yes
012474  004737  017032'                 CALL    ECHO            ;Echo character
012500  010005                          MOV     R0,R5           ;Save the converted character
012502  004737  016172'                 CALL    STRACT          ;Store the activation character
012506  000421                          BR      9$
                ;
                ;  See if we need to limit number of characters that can be
                ;  typed into a field
```

```
                                        ;
012510  016102  0000006         1$:     MOV     LFWLIM(R1),R2   ;If field width limit specified?
012514  001414                          BEQ     2$              ;Br if not
012516  026102  0000006                 CMP     LINCNT(R1),R2   ;Would this character overflow the field?
012522  103411                          BLO     2$              ;Br if not
012524  032761  0000006 0000006         BIT     #$DBGMD,LSW6(R1);Is debugger in control?
012532  001005                          BNE     2$              ;Br if yes
012534  012700  0000006                 MOV     #BELL,R0        ;Echo bell
012540  004737  017072'                 CALL    ECHO2
012544  000402                          BR      9$              ;Discard the character
                                ;
                                ;  Normal character being input
                                ;
012546  004737  015656'         2$:     CALL    INCHR           ;Store and echo the character
                                ;
                                ;  Finished
                                ;
012552  012605                  9$:     MOV     (SP)+,R5
012554  012602                          MOV     (SP)+,R2
012556  000207                          RETURN
```

```
                                        .SBTTL  DOCTRL -- Process control characters
                                ;-----------------------------------------------------------------------
                                ;  DOCTRL is called from the input interrupt character processing when
                                ;  we determine that the character being processed is a control character.
                                ;
                                ;  Inputs:
                                ;   R1 = Virtual line number.
                                ;   R5 = Character to process.
                                ;
012560                          DOCTRL:
                                ;
                                ;  See if this is a request to print the current window
                                ;
012560  120537  0000000                 CMPB    R5,VVPWCH       ;Request to print screen?
012564  001016                          BNE     1$              ;Br if not
012566  032761  0000000 0000000         BIT     #$PWKEY,LSW11(R1);Is print window control char enabled?
012574  001412                          BEQ     1$              ;Br if not
012576  016100  0000000                 MOV     LWINDO(R1),R0   ;Is windowing enabled for this process?
012602  001407                          BEQ     1$              ;Br if not -- Treat char like ordinary char
                                ;
                                ;  This is a request to print the current window contents
                                ;
012604  010246                          MOV     R2,-(SP)
012606  010002                          MOV     R0,R2           ;Get address of current window control blk
012610                                  OCALL   WINPRT          ;Print the window
012616  012602                  2$:     MOV     (SP)+,R2
012620  000404                          BR      9$
                                ;
                                ;  This is an ordinary control character
                                ;
012622  010500                  1$:     MOV     R5,R0           ;Get the control character
012624  006300                          ASL     R0              ;Convert to word table index
012626  004770  012634'                 CALL    @CTLRTN(R0)     ;Call appropriate processing routine
                                ;
                                ;  Finished
                                ;
012632  000207                  9$:     RETURN

                                ;-----------------------------------------------------------------------
                                ;  Branch table for control character processing routines.
                                ;
012634  012342'                 CTLRTN: .WORD   REGCHR           ; 00 - NUL
012636  012342'                         .WORD   REGCHR           ; 01 - SOH (control-A)
012640  012342'                         .WORD   REGCHR           ; 02 - STX (control-B)
012642  013174'                         .WORD   ICPCTC           ; 03 - ETX (control-C)
012644  013524'                         .WORD   ICPCTD           ; 04 - EOT (control-D)
012646  012342'                         .WORD   REGCHR           ; 05 - ENQ (control-E)
012650  012342'                         .WORD   REGCHR           ; 06 - ACK (control-F)
012652  013574'                         .WORD   ICPCTG           ; 07 - BEL (control-G)
012654  012342'                         .WORD   REGCHR           ; 10 - Backspace
012656  012342'                         .WORD   REGCHR           ; 11 - TAB (control-I)
012660  013070'                         .WORD   ICPLF            ; 12 - Line feed
012662  012342'                         .WORD   REGCHR           ; 13 - VT  (control-K)
012664  012342'                         .WORD   REGCHR           ; 14 - FF  (control-L)
012666  012734'                         .WORD   ICPCR            ; 15 - Carriage return
012670  012342'                         .WORD   REGCHR           ; 16 - SO  (control-N)
012672  013660'                         .WORD   ICPCTO           ; 17 - SI  (control-O)
```

```
     58 012674  012342'                    .WORD   REGCHR          ; 20 - DLE (control-P)
     59 012676  012342'                    .WORD   REGCHR          ; 21 - DC1 (control-Q)
     60 012700  013750'                    .WORD   ICPCTR          ; 22 - DC2 (control-R)
     61 012702  012342'                    .WORD   REGCHR          ; 23 - DC3 (control-S)
     62 012704  012342'                    .WORD   REGCHR          ; 24 - DC4 (control-T)
     63 012706  014014'                    .WORD   ICPCTU          ; 25 - NAK (control-U)
     64 012710  012342'                    .WORD   REGCHR          ; 26 - SYN (control-V)
     65 012712  012342'                    .WORD   REGCHR          ; 27 - ETB (control-W)
     66 012714  014374'                    .WORD   ICPCTX          ; 30 - CAN (control-X)
     67 012716  012342'                    .WORD   REGCHR          ; 31 - EM  (control-Y)
     68 012720  014460'                    .WORD   ICPCTZ          ; 32 - SUB (control-Z)
     69 012722  014514'                    .WORD   ICPESC          ; 33 - ESC
     70 012724  012342'                    .WORD   REGCHR          ; 34 - FS
     71 012726  012342'                    .WORD   REGCHR          ; 35 - GS
     72 012730  012342'                    .WORD   REGCHR          ; 36 - RS
     73 012732  012342'                    .WORD   REGCHR          ; 37 - US
```

```
                                        .SBTTL  ICPCR  -- Carriage-return processing
                                ;-----------------------------------------------------------------
                                ;  Process Carriage-return character.
                                ;
                                ;  Inputs:
                                ;   R1 = Virtual line index number.
                                ;   R5 = Current input character.
                                ;
012734  010546                  ICPCR:  MOV     R5,-(SP)
                                ;
                                ;  See if we are in single-character activation mode
                                ;
012736  004737  017254'                 CALL    SLCHK           ;Are we in single-char activation mode?
012742  103005                          BCC     1$              ;Br if not
012744  004737  016172'                 CALL    STRACT          ;Pass carriage return to program
012750  112705  000000G                 MOVB    #LF,R5          ;and follow with line feed
012754  000433                          BR      3$
                                ;
                                ;  We are not in single-character activation mode.
                                ;  See if ODT is running.
                                ;
012756  032761  000000G 000000G 1$:     BIT     #$ODTMD,LSW4(R1);Is ODT in control?
012764  001410                          BEQ     4$              ;Br if not
012766  010500                          MOV     R5,R0           ;Get the CR character
012770  004737  017032'                 CALL    ECHO            ;Echo CR
012774  012700  000000G                 MOV     #LF,R0          ;Get LF character
013000  004737  017036'                 CALL    ECHO1           ;Echo LF
013004  000417                          BR      3$              ;Store CR as activation character
                                ;
                                ;  We are not in ODT mode
                                ;
013006  004737  015656'         4$:     CALL    INCHR           ;Store and echo CR
013012  112705  000000G                 MOVB    #LF,R5          ;We will follow CR with LF
013016  032761  000000G 000000G         BIT     #$DBGMD,LSW6(R1);Is a debug program running?
013024  001004                          BNE     2$              ;Br if yes
013026  032761  000000G 000000G         BIT     #$NOLF,LSW6(R1) ;Are we suppressing LF echoing after CR?
013034  001003                          BNE     3$              ;Br if yes
013036  010500                  2$:     MOV     R5,R0           ;Get character to echo
013040  004737  017036'                 CALL    ECHO1           ;Echo line-feed
013044  004737  016172'         3$:     CALL    STRACT          ;Store activation character
                                ;
                                ;  Finished
                                ;
013050  042761  000000G 000000G 9$:     BIC     #$1CTLC,LSW5(R1);Say last character was not control-C
013056  042761  000000G 000000G         BIC     #$1ESC,LSW(R1)  ;Say last char was not escape
013064  012605                          MOV     (SP)+,R5
013066  000207                          RETURN
```

```
                                        .SBTTL  ICPLF  -- Line-feed processing
                                ;-----------------------------------------------------------------------
                                ;  Process Line-feed input character.
                                ;
                                ;  Inputs:
                                ;   R1 = Virtual line index number.
                                ;   R5 = Current input character.
                                ;
013070  010546                  ICPLF:  MOV     R5,-(SP)
                                ;
                                ;  If we are in "paper-tape" mode, ignore line-feed characters.
                                ;
013072  032761  0000000 0000000         BIT     #$TAPE,LSW2(R1) ;Are we in paper-tape mode?
013100  001025                          BNE     9$              ;Br if yes
                                ;
                                ;  See if we are in single-character activation mode
                                ;
013102  004737  017376'                 CALL    SCACHR          ;See if we are in single-char activation mode
013106  103422                          BCS     9$              ;Br if yes
                                ;
                                ;  We are not in single-character activation mode.
                                ;  See if we are in ODT mode.
                                ;
013110  032761  0000000 0000000         BIT     #$ODTMD,LSW4(R1);Is ODT in control?
013116  001412                          BEQ     2$              ;Br if not
013120  112700  0000000                 MOVB    #CR,R0          ;Echo carriage return
013124  004737  017032'                 CALL    ECHO
013130  110500                          MOVB    R5,R0           ;Get line feed character
013132  004737  017036'                 CALL    ECHO1           ;Echo line-feed
013136  004737  016172'                 CALL    STRACT          ;Store line feed and activate
013142  000404                          BR      9$
                                ;
                                ;  Treat line-feed exactly like carriage-return.
                                ;
013144  112705  0000000         2$:     MOVB    #CR,R5          ;Get CR character
013150  004737  012734'                 CALL    ICPCR           ;Process the carriage-return
                                ;
                                ;  Finished
                                ;
013154  042761  0000000 0000000 9$:     BIC     #$1CTLC,LSW5(R1);Say last character was not control-C
013162  042761  0000000 0000000         BIC     #$1ESC,LSW(R1)  ;Say last char was not escape
013170  012605                          MOV     (SP)+,R5
013172  000207                          RETURN
```

```
                                        .SBTTL  ICPCTC -- Control-C processing
                                ;-----------------------------------------------------------------------
                                ;  Process a control-C input character.
                                ;
                                ;  Inputs:
                                ;   R1 = Virtual line index number.
                                ;   R5 = Current input character.
                                ;
013174  010246                  ICPCTC: MOV     R2,-(SP)
013176  010446                          MOV     R4,-(SP)
                                ;
                                ;  Determine if we have received 2 consecutive control-C characters.
                                ;
013200  032761  000000G 000000G         BIT     #$1CTLC,LSW5(R1);Was last character control-C?
013206  001021                          BNE     2$              ;Br if yes
                                ;
                                ;  This is 1st control-C
                                ;
013210  052761  000000G 000000G         BIS     #$1CTLC,LSW5(R1);Remember last char was control-C
013216  042761  000000G 000000G 7$:     BIC     #$1ESC,LSW(R1)  ;Say last char was not escape
013224  042761  000000G 000000G         BIC     #$RTCS,LSW9(R1) ;Say we are not receiving control sequence
013232  004737  017376'                 CALL    SCACHR          ;See if we are in single-char activation mode
013236  103527                          BCS     9$              ;Br if yes
013240  004737  017110'                 CALL    ECOCTL          ;Echo "^C"
013244  004737  016172'                 CALL    STRACT          ;Store control-C as activation char
013250  000522                          BR      9$
                                ;
                                ;  We received two consecutive control-C characters
                                ;
013252  032761  000000C 000000G 2$:     BIT     #<$SUCF!$LOFCF>,LSW9(R1);Are we doing logon/logoff processing?
013260  001356                          BNE     7$              ;Br if yes -- don't allow ctrl-C abort here
013262  032761  000000G 000000G         BIT     #$SCCA,LSW5(R1) ;Suppressing control-C aborts for program?
013270  001352                          BNE     7$              ;Br if yes
013272  032761  000000G 000000G         BIT     #$NOIN,LSW3(R1) ;Special no-input flag set?
013300  001346                          BNE     7$              ;Br if yes
013302  032761  000000G 000000G         BIT     #$RFRSH,LSW4(R1);Is a window refresh being done now?
013310  001342                          BNE     7$              ;Don't allow abort during refresh
013312  016102  000000G                 MOV     LSCCA(R1),R2    ;Did user do .SCCA EMT?
013316  001412                          BEQ     3$              ;Br if not
013320  032761  000000G 000000G         BIT     #$DBGMD,LSW6(R1);Is debugger in control?
013326  001006                          BNE     3$              ;Br if yes -- ignore .SCCA
                                ;
                                ;  User did a .SCCA -- Set flag to remember to tell him about ctrl-C's
                                ;
013330  052761  000000G 000000G         BIS     #$SETCC,LSW4(R1);Remember to tell him later
013336  105237  000000G                 INCB    DOSCHD          ;Request a job scheduler cycle
013342  000725                          BR      7$
                                ;
                                ;  User did not do a .SCCA so abort him
                                ;
013344  042761  000000C 000000G 3$:     BIC     #<LCBIT!SPCTTY>,LJSW(R1) ;Clean out JSW
013352  016161  000000G 000000G         MOV     LINNXT(R1),LINPNT(R1) ;Kill all input
013360  016161  000000G 000000G         MOV     LINNXT(R1),LSTACT(R1)
013366  005061  000000G                 CLR     LINCNT(R1)
013372  016161  000000G 000000G         MOV     LINSIZ(R1),LINSPC(R1)
013400  016161  000000G 000000G         MOV     LOTSIZ(R1),LOTSPC(R1) ;Kill all output
013406  016161  000000G 000000G         MOV     LOTNXT(R1),LOTPNT(R1)
```

```
     58 013414  005061  000000G                 CLR     LFWLIM(R1)          ;No field width limit
     59 013420  005061  000000G                 CLR     LAFSIZ(R1)          ;No field width activation
     60 013424  042761  000000G 000000G         BIC     #$SLINI,LSW7(R1);Say SL must reinitialize for next line
     61 013432  042761  000000C 000000G         BIC     #<$DODFR!$GCECO>,LSW3(R1) ;Reset deferred echoing
     62 013440  016104  000000G                 MOV     LNPRIM(R1),R4       ;Get primary job number
     63 013444  042764  000000G 000000G         BIC     #$CTRLS,LSW3(R4);Reset control-S output suspension
     64 013452  004737  017110'                 CALL    ECOCTL              ;Echo "^C"
     65 013456  052761  000000G 000000G         BIS     #$CTRLC,LSW(R1) ;Set job abort flag
     66 013464  052761  000000G 000000G         BIS     #$CFKIL,LSW6(R1);Set flag to abort all open command files
     67 013472  042761  000000G 000000G         BIC     #$SUSPN,LSW(R1) ;Clear job-suspended flag
     68 013500  004737  000000G                 CALL    FORCEX              ;Force execution of the job
     69 013504  005061  000000G                 CLR     LACTIV(R1)          ;Say no chars received yet
     70 013510  042761  000000G 000000G         BIC     #$1STCH,LSW3(R1)
     71                                 ;
     72                                 ;  Finished
     73                                 ;
     74 013516  012604                  9$:     MOV     (SP)+,R4
     75 013520  012602                          MOV     (SP)+,R2
     76 013522  000207                          RETURN
```

```
                                        .SBTTL  ICPCTD -- Control-D processing
                                ;-----------------------------------------------------------------------
                                ;  Process a Control-D character.
                                ;  This forces a debugger breakpoint if we are running under the debugger.
                                ;
                                ;  Inputs:
                                ;   R1 = Virtual line index number.
                                ;   R5 = Current input character.
                                ;
013524  032761  0000000G 0000000G ICPCTD: BIT     #$INKMN,LSW4(R1);Are we running in TSKMON now?
013532  001405                           BEQ     2$              ;Br if not
013534  032761  0000000G 0000000G         BIT     #$DBKMN,LSW9(R1);Should we debug TSKMON?
013542  001411                           BEQ     1$              ;Br if not
013544  000404                           BR      3$              ;Yes, go force breakpoint
013546  032761  000000C 0000000G 2$:     BIT     #$DEBUG!$CTRLD,LSW9(R1);Is job running with the debugger?
013554  001404                           BEQ     1$              ;Br if not
013556  052761  0000000G 0000000G 3$:     BIS     #$DBGBK,LSW9(R1);Set flag to force debug breakpoint
013564  000402                           BR      9$
                                ;
                                ;  We are not running with debugger, treat Ctrl-D like normal character
                                ;
013566  004737  012342'          1$:      CALL    REGCHR          ;Store as regular character
                                ;
                                ;  Finished
                                ;
013572  000207                  9$:      RETURN
```

```
                                        .SBTTL  ICPCTG -- Control-G processing
                                ;-----------------------------------------------------------------------
                                ;  Process a Control-G (Bell) character.
                                ;
                                ;  Inputs:
                                ;   R1 = Virtual line index number.
                                ;   R5 = Current input character.
                                ;
013574  032761  000000G 000000G ICPCTG: BIT     #SPCTTY,LJSW(R1);Is special character processing wanted?
013602  001415                         BEQ     2$              ;Br if not
013604  032761  000000G 000000G         BIT     #$CHACT,LSW5(R1);Is single character activation wanted?
013612  001403                         BEQ     1$              ;Br if not
013614  004737  016252'                CALL    STRSNG          ;Store the character
013620  000410                         BR      9$
                                ;
                                ;  Special-TTY mode is set but not single character activation
                                ;
013622  010500                 1$:     MOV     R5,R0           ;Get char to echo
013624  004737  017032'                CALL    ECHO            ;Echo a bell
013630  004737  016172'                CALL    STRACT          ;Store and activate
013634  000402                         BR      9$
                                ;
                                ;  Normal character processing
                                ;
013636  004737  012342'         2$:     CALL    REGCHR          ;Treat as regular character
                                ;
                                ;  Finished
                                ;
013642  042761  0000000 0000000 9$:     BIC     #$1CTLC,LSW5(R1);Say last character was not control-C
013650  042761  0000000 0000000         BIC     #$1ESC,LSW(R1)  ;Say last char was not escape
013656  000207                         RETURN
```

```
                                        .SBTTL  ICPCTO -- Control-O processing
                                ;-----------------------------------------------------------------------
                                ;  Process a Control-O input character.
                                ;
                                ;  Inputs:
                                ;   R1 = Virtual line index number.
                                ;   R5 = Current input character.
                                ;
013660  032761  000000G 000000G ICPCTO: BIT     #$CTRLO,LSW3(R1);Is output currently suppressed?
013666  001022                          BNE     1$              ;Br if yes (reenable it)
                                ;
                                ;  Begin suppressing output
                                ;
013670  016161  000000G 000000G         MOV     LOTSIZ(R1),LOTSPC(R1) ;Empty output buffer
013676  016161  000000G 000000G         MOV     LOTNXT(R1),LOTPNT(R1)
013704  004737  017110'                 CALL    ECOCTL          ;Echo "^O"
013710  052761  000000G 000000G         BIS     #$CTRLO,LSW3(R1);Remember output is being suppressed
013716  026127  000000G 000000G         CMP     LSTATE(R1),#S$OTWT ;Is job waiting for output space?
013724  001010                          BNE     2$              ;Br if not
013726  004737  005744'                 CALL    OTREGO          ;Reactivate the job
013732  000405                          BR      2$
                                ;
                                ;  Reenable output
                                ;
013734  042761  000000G 000000G 1$:     BIC     #$CTRLO,LSW3(R1);Remove output suppression
013742  004737  017110'                 CALL    ECOCTL          ;Echo "^O"
                                ;
                                ;  Finished
                                ;
013746  000207                  2$:     RETURN
```

```
                                        .SBTTL  ICPCTR -- Control-R processing
                                ;-----------------------------------------------------------------------
                                ;  Process control-R input character.
                                ;
                                ;  Inputs:
                                ;   R1 = Virtual line index number.
                                ;   R5 = Current input character.
                                ;
013750  010246                  ICPCTR: MOV     R2,-(SP)
013752  004737  0000000G                CALL    BRKPT           ;Call TSEXEC for debugging breakpoint
013756  004737  017376'                CALL    SCACHR          ;Is job in single-char activation mode?
013762  103412                         BCS     9$              ;Br if yes
                                ;
                                ;  Redisplay the current input line
                                ;
013764  004737  017110'         4$:    CALL    ECOCTL          ;Echo "^R"
013770  016102  0000000G               MOV     LSTACT(R1),R2   ;Point past last activation char
013774  004737  016312'         1$:    CALL    FETCHR          ;Get next char from TT input buffer
014000  103403                         BCS     9$              ;Br if no more characters
014002  004737  017036'               CALL    ECHO1           ;Echo the character
014006  000772                         BR      1$              ;Continue printing line
                                ;
                                ;  Finished
                                ;
014010  012602                  9$:    MOV     (SP)+,R2
014012  000207                         RETURN
```

```
                                        .SBTTL  ICPCTU -- Control-U processing
                                ;-----------------------------------------------------------------------
                                ;  Process a Control-U character.
                                ;
                                ;  Inputs:
                                ;   R1 = Virtual line index number.
                                ;   R5 = Current input character.
                                ;
014014  010246                  ICPCTU: MOV     R2,-(SP)
014016  010346                          MOV     R3,-(SP)
                                ;
                                ;  Remember that last character was not escape or control-C
                                ;
014020  042761  000000G 000000G         BIC     #$1CTLC,LSW5(R1);Last character was not control-C
014026  042761  000000G 000000G         BIC     #$1ESC,LSW(R1)  ;Say last char was not escape
                                ;
                                ;  Determine if we are in single-character activation mode
                                ;
014034  004737  017376'                 CALL    SCACHR          ;See if we are in single-char activation mode
014040  103545                          BCS     27$             ;Br if yes
                                ;
                                ;  We are not in single-charactr activation mode
                                ;
014042  042761  000000G 000000G         BIC     #$RBOUT,LSW3(R1);Reset rubout mode
014050  032761  000000G 000000G         BIT     #$DODFR,LSW3(R1);Doing deferred echoing?
014056  001133                          BNE     5$              ;Br if yes
                                ;
                                ;  Determine if this is a scope type terminal
                                ;
014060  032761  000000G 000000G         BIT     #$SCOPE,LSW2(R1);Is this a scope terminal?
014066  001003                          BNE     15$             ;Br if yes
                                ;
                                ;  Echo "^U" for non-scope terminals
                                ;
014070  004737  017110'                 CALL    ECOCTL          ;Echo "^U"
014074  000524                          BR      5$              ;Go delete the characters
                                ;
                                ;  Do line erase for scope terminals
                                ;
014076  032761  000000G 000000G 15$:    BIT     #$1STCH,LSW3(R1);Any characters received yet?
014104  001523                          BEQ     27$             ;Br if not
014106  116103  000000G                 MOVB    LCOL(R1),R3     ;Get current column position
014112  042703  177400                  BIC     #^C377,R3       ;Kill sign extension
014116  116102  000000G                 MOVB    LINCUR(R1),R2   ;Get start of line position
014122  042702  177400                  BIC     #^C377,R2       ;Kill sign extension
014126  160203                          SUB     R2,R3           ;Get # of columns in field being erased
014130  003506                          BLE     5$              ;Br if nothing to erase
014132  126127  000000G 000000G         CMPB    LRBFIL(R1),#SPACE; Is rubout filler char = space?
014140  001056                          BNE     8$              ;Br if not
014142  032761  000000C 000000G         BIT     #VT52!VT100!VT2007!VT2008!HAZEL,LTRMTP(R1) ;Can we line erase?
014150  001452                          BEQ     8$              ;Br if not
014152  005702                          TST     R2              ;Are we erasing entire line?
014154  001005                          BNE     10$             ;Br if not
                                ;
                                ;  Going to start of line.  Use carriage-return to get there.
                                ;
014156  112700  000000G                 MOVB    #CR,R0          ;Send carriage return to terminal
```

```
     58 014162  004737  003122'                    CALL    PUTCH2
     59 014166  000405                             BR      11$
     60                                 ;
     61                                 ;  Not going to start of line.  Use backspaces to get to start of field.
     62                                 ;
     63 014170  112700  000000G         10$:    MOVB    #BKSPAC,R0      ;Get backspace character
     64 014174  004737  017036'         12$:    CALL    ECHO1           ;Send backspace to terminal
     65 014200  077303                          SOB     R3,12$          ;Send enough to get to start of field
     66                                 ;
     67                                 ;  Use scope control sequence to erase the line
     68                                 ;
     69 014202  012703  014362'         11$:    MOV     #ERV52,R3       ;Assume this is a VT52 terminal
     70 014206  116100  000000G                 MOVB    LNPRIM(R1),R0   ;Get primary job index number
     71 014212  032760  000000G 000000G        BIT     #$V52EM,LSW11(R0);Are we emulating a VT52?
     72 014220  001014                          BNE     13$             ;Br if yes
     73 014222  016100  000000G                 MOV     LTRMTP(R1),R0   ;Get terminal type code
     74 014226  032700  000000G                 BIT     #VT52,R0        ;Is this a VT52 terminal?
     75 014232  001007                          BNE     13$             ;Br if yes
     76 014234  012703  014371'                 MOV     #ERHAZL,R3      ;Assume hazeltine terminal
     77 014240  032700  000000G                 BIT     #HAZEL,R0       ;Is this a Hazeltine terminal?
     78 014244  001002                          BNE     13$             ;Br if yes
     79 014246  012703  014365'                 MOV     #ERV100,R3      ;Assume this is a VT100 or VT200
     80 014252  112300                  13$:    MOVB    (R3)+,R0        ;Get next char from control sequence
     81 014254  001434                          BEQ     5$              ;Br if end reached
     82 014256  120027  000037                  CMPB    R0,#37          ;Is this a control or printing character
     83 014262  101402                          BLOS    14$             ;Br if control character
     84 014264  105361  000000G                 DECB    LCOL(R1)        ;Correct column counter if sending printing ch
     85 014270  004737  003122'         14$:    CALL    PUTCH2          ;Send character to terminal
     86 014274  000766                          BR      13$             ;Continue sending control sequence
     87                                 ;
     88                                 ;  Kill all input characters by sending
     89                                 ;  backspace-space-backspace...
     90                                 ;
     91 014276  116102  000000G         8$:     MOVB    LRBFIL(R1),R2   ;Get rubout-filler character
     92 014302  032761  000000G 000000G        BIT     #$DBGMD,LSW6(R1);Is debugger doing I/O now?
     93 014310  001402                          BEQ     9$              ;Br if not
     94 014312  112702  000040                  MOVB    #' ,R2          ;Use blank for debugger rubout
     95 014316  112700  000000G         9$:     MOVB    #BKSPAC,R0
     96 014322  004737  017036'                 CALL    ECHO1           ;Send backspace
     97 014326  110200                          MOVB    R2,R0           ;Get rubout-filler character
     98 014330  004737  017036'                 CALL    ECHO1           ;Send filler character
     99 014334  112700  000000G                 MOVB    #BKSPAC,R0      ;Send backspace
    100 014340  004737  017036'                 CALL    ECHO1
    101 014344  077314                          SOB     R3,9$           ;Loop on number of characters to kill
    102                                 ;
    103                                 ;  Now delete characters from the input buffer
    104                                 ;
    105 014346  004737  015564'         5$:     CALL    KILCHR          ;Kill last character in the buffer
    106 014352  103375                          BCC     5$              ;Loop if more left to kill
    107                                 ;
    108                                 ;  Finished
    109                                 ;
    110 014354  012603                  27$:    MOV     (SP)+,R3
    111 014356  012602                          MOV     (SP)+,R2
    112 014360  000207                          RETURN
    113                                 ;
    114                                 ;  Terminal control sequences to erase a line.
```

```
    115                                         ;
    116 014362      0000    113     000 ERV52:  .BYTE   ESC,113,0       ;VT52
    117 014365      0000    133     113 ERV100: .BYTE   ESC,133,113,0   ;VT100 and VT200
        014370      000
    118 014371      176     017     000 ERHAZL: .BYTE   176,17,0         ;Hazeltine
    119                                         .EVEN
```

```
                                        .SBTTL  ICPCTX -- Control-X processing
                                ;-----------------------------------------------------------------------
                                ;  Process a Control-X character.
                                ;
                                ;  Inputs:
                                ;   R1 = Virtual line index number.
                                ;   R5 = Current input character.
                                ;
014374  032761  0000000 0000000 ICPCTX: BIT     #SPCTTY,LJSW(R1);Is special character processing wanted?
014402  001415                          BEQ     2$              ;Br if not
014404  032761  0000000 0000000         BIT     #$CHACT,LSW5(R1);Is single character activation wanted?
014412  001403                          BEQ     1$              ;Br if not
014414  004737  016252'                CALL    STRSNG          ;Store the character
014420  000410                          BR      9$
                                ;
                                ;  Special-TTY mode is set but not single character activation
                                ;
014422  010500                  1$:     MOV     R5,R0           ;Get char to echo
014424  004737  017110'                CALL    ECOCTL          ;Echo"^X"
014430  004737  016172'                CALL    STRACT          ;Store the character and activate
014434  000402                          BR      9$
                                ;
                                ;  Normal character processing
                                ;
014436  004737  012342'          2$:    CALL    REGCHR          ;Treat as regular character
                                ;
                                ;  Finished
                                ;
014442  042761  0000000 0000000 9$:     BIC     #$1CTLC,LSW5(R1);Say last character was not control-C
014450  042761  0000000 0000000         BIC     #$1ESC,LSW(R1)  ;Say last char was not escape
014456  000207                          RETURN
```

```
      1                                       .SBTTL  ICPCTZ -- Control-Z processing
      2                                ;-----------------------------------------------------------------------
      3                                ;  Process Control-Z character.
      4                                ;
      5                                ;  Inputs:
      6                                ;   R1 = Virtual line index number.
      7                                ;   R5 = Current input character.
      8                                ;
      9 014460  004737  017376'        ICPCTZ: CALL    SCACHR          ;Are we in single character activation mode?
     10 014464  103404                         BCS     9$              ;Br if yes
     11 014466  004737  017110'                CALL    ECOCTL          ;Echo "^Z"
     12 014472  004737  016172'                CALL    STRACT          ;Store character and activate
     13                               ;
     14                               ;  Say last character was not escape or control-C
     15                               ;
     16 014476  042761  000000@ 000000@ 9$:     BIC     #$1CTLC,LSW5(R1);Say last character was not control-C
     17 014504  042761  000000@ 000000@        BIC     #$1ESC,LSW(R1)  ;Say last char was not escape
     18 014512  000207                        RETURN
```

```
                                        .SBTTL  ICPESC -- Escape processing
                                ;-----------------------------------------------------------------------
                                ;  Process an escape character.
                                ;
                                ;  Inputs:
                                ;   R1 = Virtual line index number.
                                ;   R5 = Current input character.
                                ;
014514  004737  017254'         ICPESC: CALL    SLCHK           ;See if SL should get escape
014520  103410                          BCS     4$              ;Br if SL wants characters
014522  032761  0000000@ 0000000@       BIT     #SPCTTY,LJSW(R1);Is job in special TTY mode?
014530  001407                          BEQ     1$              ;Br if not
014532  032761  0000000@ 0000000@       BIT     #$CHACT,LSW5(R1);Is single char activation enabled?
014540  001412                          BEQ     2$              ;Br if not
014542  004737  016252'         4$:     CALL    STRSNG          ;Store and activate
014546  000432                          BR      9$
                                ;
                                ;  Escape and not single character activation
                                ;
014550  112700  000044          1$:     MOVB    #'$,R0          ;Echo "$"
014554  004737  017032'                  CALL    ECHO
014560  004737  015674'                  CALL    STRCHR          ;Store escape as non-activation char
014564  000423                          BR      9$
                                ;
                                ;  Program is in special-TTY mode but single character activation
                                ;  is not enabled.
                                ;
014566  112700  000044          2$:     MOVB    #'$,R0          ;Echo "$"
014572  004737  017032'                  CALL    ECHO
014576  032761  0000000@ 0000000@       BIT     #$1ESC,LSW(R1)  ;Was last character ESC?
014604  001006                          BNE     3$              ;Br if yes
014606  004737  015674'                  CALL    STRCHR          ;Store the escape
014612  052761  0000000@ 0000000@       BIS     #$1ESC,LSW(R1)  ;Remember that last char is escape
014620  000405                          BR      9$
014622  004737  016172'         3$:     CALL    STRACT          ;Store escape and activate
014626  042761  0000000@ 0000000@       BIC     #$1ESC,LSW(R1)  ;Say last char was not escape
                                ;
                                ;  Finished
                                ;
014634  042761  0000000@ 0000000@ 9$:     BIC     #$1CTLC,LSW5(R1);Say last character was not control-C
014642  000207                          RETURN
```

```
                                        .SBTTL  ICPRUB -- Rubout processing
                                ;-----------------------------------------------------------------------
                                ;  Process a rubout character.
                                ;
                                ;  Inputs:
                                ;   R1 = Virtual line number.
                                ;   R5 = Rubout character.
                                ;
014644  010246                  ICPRUB: MOV     R2,-(SP)
014646  010346                          MOV     R3,-(SP)
                                ;
                                ;  Remember that last character was not control-C
                                ;
014650  042761  000000G 000000G         BIC     #$1CTLC,LSW5(R1);Last character was not control-C
014656  042761  000000G 000000G         BIC     #$1ESC,LSW(R1)  ;Say last char was not escape
                                ;
                                ;  See if we are in single-character activation mode
                                ;
014664  004737  017376'                 CALL    SCACHR          ;Are we in single-char activation mode
014670  103541                          BCS     2$              ;Br if yes
                                ;
                                ;  We are not in single character activation mode.
                                ;
014672  032761  000000G 000000G         BIT     #$ECHO,LSW2(R1) ;Is echoing turned on?
014700  001004                          BNE     12$             ;Br if yes
014702  032761  000000G 000000G         BIT     #$DBGMD,LSW6(R1);Is debugger in control?
014710  001417                          BEQ     30$             ;Br if not
014712  032761  000000G 000000G 12$:    BIT     #$SCOPE,LSW2(R1);Scope type terminal?
014720  001021                          BNE     27$             ;Br if yes
                                ;
                                ;  Rubout on non-scope terminal.
                                ;  Echo \xxx\ type rubout sequence.
                                ;
014722  032761  000000G 000000G         BIT     #$RBOUT,LSW3(R1);Are we already doing rubout sequence?
014730  001007                          BNE     30$             ;Br if yes
014732  112700  000134                 MOVB    #'\,R0          ;Get backslash to begin rubout sequence
014736  004737  017036'                 CALL    ECHO1           ;Echo "\"
014742  052761  000000G 000000G         BIS     #$RBOUT,LSW3(R1);Say we have started rubout sequence
014750  004737  015564'         30$:    CALL    KILCHR          ;Delete one character from buffer
014754  103507                          BCS     2$              ;Br if no char to delete
014756  004737  017036'                 CALL    ECHO1           ;Echo deleted char
014762  000504                          BR      2$
                                ;
                                ;  Do backspace editing for scope type terminals
                                ;
014764  004737  015564'         27$:    CALL    KILCHR          ;Delete one character from buffer
014770  103501                          BCS     2$              ;Br if no character was left to delete
014772  120027  000000G                 CMPB    R0,#TAB         ;Did we delete a tab character?
014776  001044                          BNE     10$
                                ;
                                ;  We just deleted a tab character.
                                ;  Do correct backspacing.
                                ;
015000  116103  000000G                 MOVB    LINCUR(R1),R3   ;Position of start of line
015004  042703  177400                  BIC     #^C377,R3       ;Kill sign extension
015010  016102  000000G                 MOV     LSTACT(R1),R2   ;Start of input string
015014  004737  016312'         3$:     CALL    FETCHR          ;Get next char from TT input buffer
```

```
                        BCS     6$              ;Br if no more characters
                        CMPB    R0,#TAB         ;Is character a tab?
                        BNE     7$              ;Br if not
                        ADD     #8.,R3          ;Calculate spaces over it
                        BIC     #7,R3
                        BR      3$
                7$:     CMPB    R0,#BKSPAC      ;Was character backspace?
                        BNE     5$              ;Br if not
                        DEC     R3              ;Backup cursor position
                        BR      3$
                5$:     INC     R3              ;Advance cursor for normal character
                        BR      3$
                6$:     MOV     R3,R2           ;Save position in front of last tab
                        ADD     #8.,R2          ;Calc position after tab
                        BIC     #7,R2
                        SUB     R3,R2           ;Calc number of backspace needed
                        MOV     #BKSPAC,R0      ;Now backspace cursor
                9$:     CALL    ECHO1           ;Backspace over tab columns
                        SOB     R2,9$
                        BR      2$
                ;
                ;  Check for rubout of backspace character
                ;
                10$:    CMPB    R0,#BKSPAC      ;Is deleted character a backspace?
                        BNE     11$             ;Br if not
                        MOVB    #SPACE,R0       ;Output a space to kill the backspace
                        CALL    ECHO1
                        BR      2$
                ;
                ;  Rubout of regular character.
                ;  Echo backspace-space-backspace
                ;
                11$:    MOVB    #BKSPAC,R0      ;Echo backspace
                        CALL    ECHO1
                        MOVB    LRBFIL(R1),R0   ;Get rubout-filler character
                        BIT     #$DBGMD,LSW6(R1);Is a debugger running?
                        BEQ     13$             ;Br if not
                        MOVB    #' ,R0          ;Use space for debugger rubout
                13$:    CALL    ECHO1           ;Echo rubout-filler character
                        MOVB    #BKSPAC,R0      ;Now echo backspace
                        CALL    ECHO1
                ;
                ;  Finished
                ;
                2$:     MOV     (SP)+,R3
                        MOV     (SP)+,R2
                        RETURN
```

```
                                        .SBTTL  CKVTAC -- Check for VTxx escape-letter activation
                                ;-----------------------------------------------------------------------
                                ;  We are activating on VTxx escape-letter sequences.
                                ;  See if this is one and we should activate.
                                ;
                                ;  Inputs:
                                ;   R1 = Virtual line index number.
                                ;   R5 = Current input character.
                                ;
                                ;  Outputs:
                                ;   C-flag cleared ==> Character was totally processed here.
                                ;   C-flag set     ==> Character was not processed by us.
                                ;
015202                          CKVTAC:
                                ;
                                ;  See if this character is ESC, CSI, or SS3 which starts a terminal
                                ;  control sequence.
                                ;
015202  120527  000033                  CMPB    R5,#33          ;Is character ESC?
015206  001007                          BNE     1$              ;Br if not
015210  052761  0000000 0000000        BIS     #$1ESC,LSW(R1)  ;Remember last char was an escape
015216  052761  0000000 0000000        BIS     #$RTCS,LSW9(R1) ;Say we are receiving a control sequence
015224  000450                          BR      15$             ;Char is part of sequence
015226  120527  000233          1$:     CMPB    R5,#233         ;Is character CSI?
015232  001403                          BEQ     14$             ;Br if yes
015234  120527  000217                  CMPB    R5,#217         ;Is character SS3?
015240  001004                          BNE     2$              ;Br if not
                                ;
                                ;  This character begins a new control sequence
                                ;
015242  052761  0000000 0000000 14$:     BIS     #$RTCS,LSW9(R1) ;Say we are receiving a control sequence
015250  000433                          BR      3$              ;Go store the character
                                ;
                                ;  See if this character is part of a terminal control sequence
                                ;
015252  032761  0000000 0000000 2$:      BIT     #$RTCS,LSW9(R1) ;Are we currently receiving a control seq?
015260  001453                          BEQ     8$              ;Br if not
                                ;
                                ;  We are currently receiving a terminal control sequence.
                                ;  See if this character terminates the sequence.
                                ;
015262  120527  000101                  CMPB    R5,#'A          ;Is this a upper-case letter?
015266  103424                          BLO     3$              ;Br if not
015270  120527  000132                  CMPB    R5,#'Z
015274  101010                          BHI     7$              ;Br if not upper-case letter
015276  120527  000117                  CMPB    R5,#'O          ;Letter O?
015302  001013                          BNE     4$              ;Br if not
015304  032761  0000000 0000000        BIT     #$1ESC,LSW(R1)  ;Was last character ESC?
015312  001407                          BEQ     4$              ;Br if not
015314  000411                          BR      3$              ;ESC O is equivalent to SS3
015316  120527  000141          7$:     CMPB    R5,#141         ;Is this a lower-case letter?
015322  103406                          BLO     3$              ;Br if not
015324  120527  000176                  CMPB    R5,#176         ;Lower case letter or ~
015330  101003                          BHI     3$              ;Br if not
                                ;
                                ;  This character terminates the control sequence
                                ;
```

```
      58 015332  042761  000000G 000000G 4$:     BIC     #$RTCS,LSW9(R1) ;Say no longer receiving control sequence
      59                                 ;
      60                                 ;  This character is part of control sequence -- Pass to program
      61                                 ;
      62 015340  042761  000000G 000000G 3$:     BIC     #$1ESC,LSW(R1)  ;Say last char was not escape
      63 015346  010546                  15$:    MOV     R5,-(SP)        ;Save the character
      64 015350  012705  000000G                 MOV     #ESCFLG,R5      ;Get char that says escape sequence follows
      65 015354  004737  015674'                 CALL    STRCHR          ;Store flag character
      66 015360  012605                          MOV     (SP)+,R5        ;Recover real character
      67 015362  032761  000000G 000000G        BIT     #$RTCS,LSW9(R1) ;Have we terminated the sequence?
      68 015370  001403                          BEQ     5$              ;Br if yes
      69 015372  004737  015674'                 CALL    STRCHR          ;Store char that is part of sequence
      70 015376  000402                          BR      6$
      71 015400  004737  016172'          5$:    CALL    STRACT          ;Store char as activation character
      72 015404  000241                  6$:     CLC                     ;Say we finished processing the character
      73 015406  000401                          BR      9$
      74                                 ;
      75                                 ;  This character is not part of a control sequence
      76                                 ;
      77 015410  000261                  8$:     SEC                     ;Signal char is not part of sequence
      78                                 ;
      79                                 ;  Finished
      80                                 ;
      81 015412  000207                  9$:     RETURN
```

```
                                        .SBTTL  CHKODT -- Check for ODT activation characters
                                ;-----------------------------------------------------------------
                                ;  CHKODT is called to determine whether ODT character
                                ;  activation is desired and whether the character in R0
                                ;  is an ODT activation character.
                                ;  When called R0 must contain the character to be tested
                                ;  and R1 must contain the user index number.
                                ;  On return the c-flag is set if the character is an
                                ;  ODT activation character.
                                ;  All registers are preserved.
                                ;
015414  032761  0000000 0000000 CHKODT:  BIT     #$ODTMD,LSW4(R1);ODT ACTIVATION DESIRED?
015422  001417                          BEQ     1$              ;BRANCH IF NOT
                                ;
                                ;  User does want odt activation.
                                ;  Is this an ODT activation char?
                                ;
015424  120027  000054                    CMPB    R0,#',          ;DON'T ACTIVATE ON ','
015430  001414                          BEQ     1$
015432  120027  000044                    CMPB    R0,#'$          ;OR '$'
015436  001411                          BEQ     1$
015440  120027  000073                    CMPB    R0,#';          ;OR ';'
015444  001406                          BEQ     1$
015446  120027  000060                    CMPB    R0,#'0          ;IS THIS A DIGIT?
015452  103405                          BLO     2$              ;BRANCH IF NOT DIGIT
015454  120027  000071                    CMPB    R0,#'9
015460  101002                          BHI     2$              ;BRANCH IF NOT DIGIT
                                ;
                                ;  Not activation character.
                                ;
1$:     015462  000241                  CLC                     ;CLEAR C-FLAG
        015464  000207                  RETURN
                                ;
                                ;  Got activation char.
                                ;
2$:     015466  000261                  SEC                     ;SET C-FLAG
        015470  000207                  RETURN
```

```
                                        .SBTTL  INFIN  -- TT input wait completed
                                ;-----------------------------------------------------
                                ;  INFIN -- Routine to add the user whose line index # is
                                ;  in R1 to the end of the TTFN queue if the user is
                                ;  currently waiting for input.
                                ;
                                ;  Inputs:
                                ;   R1 = Virtual line number of job being activated.
                                ;
015472  010246                  INFIN:  MOV     R2,-(SP)
                                ;
                                ;  Determine which state to put job in
                                ;
015474  004737  017254'                 CALL    SLCHK           ;Is program doing single character activation?
015500  103003                         BCC     2$              ;Br if not
015502  012700  0000000                MOV     #S$TTSC,R0      ;If yes, put job in special high-prio state
015506  000402                         BR      3$
015510  012700  0000000          2$:    MOV     #S$TTFN,R0      ;Put user in normal input-done state
                                ;
                                ;  If job is already in at least this high a priority state, leave it alone
                                ;
015514  016102  0000000          3$:    MOV     LSTATE(R1),R2   ;Get job's current execution state
015520  020200                         CMP     R2,R0           ;Is job already at that high a priority?
015522  101416                         BLOS    9$              ;Br if yes -- Don't change its priority
                                ;
                                ;  Job's current execution priority is lower than that associated
                                ;  with receiving an activation character.
                                ;  If job is currently waiting for an activation character, boost its
                                ;  priority to get it running again.
                                ;
015524  020227  0000000                CMP     R2,#S$INWT      ;Is job waiting for TT input?
015530  001003                         BNE     4$              ;Br if not
015532  005061  0000000                CLR     LRDTIM(R1)      ;Clear TT read time-out counter
015536  000406                         BR      1$              ;Go boost priority of job to get it running
                                ;
                                ;  Job is not waiting for TT input.
                                ;  If job is in any other wait state, leave it alone.
                                ;
015540  020227  0000000          4$:    CMP     R2,#S$$RUN      ;Is job in a wait state?
015544  103005                         BHIS    9$              ;Br if yes -- Leave it alone
                                ;
                                ;  Job is not in a wait state.
                                ;  If job is still classified as "interactive", give it a prio boost.
                                ;
015546  005761  0000000                TST     LITIME(R1)      ;Is job still interactive?
015552  001402                         BEQ     9$              ;Br if not
                                ;
                                ;  Boost priority of job
                                ;
015554  004737  0000000          1$:    CALL    ENQTL           ;Requeue job at tail of high-prio queue
                                ;
                                ;  Finished
                                ;
015560  012602                 9$:    MOV     (SP)+,R2
015562  000207                         RETURN
```

```
                                        .SBTTL  KILCHR -- Delete a character from input buffer
                                ;-----------------------------------------------------------
                                ;  KILCHR is called to delete a character
                                ;  from the input buffer.
                                ;
                                ;  Inputs:
                                ;   R1 = user index #.
                                ;
                                ;  Outputs:
                                ;   The C-flag is set on attempt to delete activation char.
                                ;   R0 = The character that was deleted.
                                ;
015564  010246                  KILCHR: MOV     R2,-(SP)
015566  010346                          MOV     R3,-(SP)
                                ;
                                ;  See if there are any characters in the TT input ring buffer
                                ;
015570  005761  0000000                 TST     LINCNT(R1)      ;ANY CHARS IN BUFFER NOW?
015574  001424                          BEQ     3$              ;IF NOT NOTHING TO DELETE
                                ;
                                ;  Locate the last character in the TT input buffer
                                ;
015576  016102  0000000                 MOV     LINNXT(R1),R2   ;Get pointer past last char in buffer
015602  005302                          DEC     R2              ;Point to last char in buffer
015604  020261  0000000                 CMP     R2,LINBUF(R1)   ;Did we go past front of buffer?
015610  103003                          BHIS    1$              ;Br if not
015612  016102  0000000                 MOV     LINEND(R1),R2   ;Get pointer past right end of buffer
015616  005302                          DEC     R2              ;Get pointer to last char in buffer
015620  010203                  1$:     MOV     R2,R3           ;Save pointer to character being deleted
015622  004737  016312'                 CALL    FETCHR          ;Get the last character in the buffer
                                ;
                                ;  Delete the last character unless it is an activation character
                                ;
015626  032700  0000000                 BIT     #ACFLAG,R0      ;Is this character an activation char?
015632  001005                          BNE     3$              ;Br if yes
015634  010302                          MOV     R3,R2           ;Get pointer to character to delete
015636  004737  016604'                 CALL    DELCHR          ;Delete the character
                                ;
                                ;  We successfully deleted a character
                                ;
015642  000241                  2$:     CLC                     ;SAY CHAR WAS DELETED
015644  000401                          BR      9$              ;Finished
                                ;
                                ;  There are no characters to delete
                                ;
015646  000261                  3$:     SEC                     ;Signal that there are no chars to delete
                                ;
                                ;  Finished
                                ;
015650  012603                  9$:     MOV     (SP)+,R3
015652  012602                          MOV     (SP)+,R2
015654  000207                          RETURN
```

```
      1                                       .SBTTL  INCHR  -- Store and echo a character
      2                                 ;-----------------------------------------------------
      3                                 ;  INCHR is called to store and echo the
      4                                 ;  input character in R5.
      5                                 ;  the user's index # must be in R1.
      6                                 ;
      7 015656  004737  015674'         INCHR:  CALL    STRCHR          ;STORE THE CHARACTER
      8 015662  103403                          BCS     1$              ;BR IF INPUT BUFFER OVERFLOW
      9 015664  010500                          MOV     R5,R0           ;GET CHARACTER INTO R0 FOR ECHO
     10 015666  004737  017032'                 CALL    ECHO            ;ECHO THE CHARACTER
     11 015672  000207                  1$:     RETURN
```

```
                                        .SBTTL  STRCHR -- Store a character into TI buffer
                                ;-----------------------------------------------------------
                                ;  STRCHR is called to store the character
                                ;  in R5 into the input buffer.
                                ;  The C-flag is set if the buffer overflows.
                                ;  R1 = User index number.
                                ;  All registers are preserved.
                                ;
015674  010046                  STRCHR: MOV     R0,-(SP)
015676  010246                          MOV     R2,-(SP)
015700  010346                          MOV     R3,-(SP)
                                ;
                                ;  Keep track of cursor position of 1st character on the line
                                ;
015702  032761  000000C 000000G         BIT     #<$1STCH!$DODFR>,LSW3(R1);1ST CHAR AFTER ACTIVATION?
015710  001006                          BNE     4$                      ;BRANCH IF NOT FIRST
015712  052761  000000G 000000G         BIS     #$1STCH,LSW3(R1);SAY WE'VE GOT A CHAR
015720  116161  000000G 000000G         MOVB    LCOL(R1),LINCUR(R1);SAVE CURSOR POSITION
                                ;
                                ;  See if buffer is about to overflow
                                ;
015726  016103  000000G         4$:     MOV     LINSPC(R1),R3   ;Get # free bytes in TT input ring buffer
015732  020327  000012                  CMP     R3,#10.         ;Got at least 10 free bytes?
015736  103022                          BHIS    7$              ;Br if yes
015740  005761  000000G                 TST     LACTIV(R1)      ;Gotten an activation char yet?
015744  001403                          BEQ     8$              ;Br if not
015746  052761  000000G 000000G         BIS     #$XSTOP,LSW6(R1);Don't move any more chars out of silo
015754  020327  000006         8$:     CMP     R3,#6.          ;Got at least 6 bytes left?
015760  103011                          BHIS    7$              ;Br if yes -- accept the character
015762  020327  000002                  CMP     R3,#2.          ;Is the buffer as full as we will allow?
015766  101470                          BLOS    5$              ;Br if yes -- Discard the character
015770  120527  000000G                 CMPB    R5,#CR          ;Is this a carriage return?
015774  001403                          BEQ     7$              ;Br if yes -- accept it
015776  032705  000000G                 BIT     #ACFLAG,R5      ;Is this an activation character?
016002  001462                          BEQ     5$              ;Br if not -- Reserve last 4 bytes for act chr
                                ;
                                ;  Determine position where char is to be stored in input ring buffer
                                ;  and update pointer to next free position.
                                ;
016004                          7$:     DISABL                  ;;;Disable interrupts
016012  016102  000000G                 MOV     LINNXT(R1),R2   ;;;Get pointer to next position in TT buffer
016016  010203                          MOV     R2,R3           ;;;
016020  005203                          INC     R3              ;;;Compute address of next char position
016022  020361  000000G                 CMP     R3,LINEND(R1)   ;;;Did we go past end of buffer?
016026  103402                          BLO     1$              ;;;Br if not
016030  016103  000000G                 MOV     LINBUF(R1),R3   ;;;Wrap around to front of buffer
016034  010361  000000G         1$:     MOV     R3,LINNXT(R1)   ;;;Save pointer to where next char goes
016040                                  ENABL                   ;Enable interrupts
                                ;
                                ;  Store character into buffer
                                ;
016046  010500                          MOV     R5,R0           ;Get character to store
016050  004737  016454'                 CALL    INSCHR          ;Store the character
                                ;
                                ;  Update character counts.
                                ;
016054  005361  000000G                 DEC     LINSPC(R1)      ;Reduce free space count for buffer
```

```
     58 016060  005261  000000G                    INC     LINCNT(R1)      ;Count another char in input buffer
     59                                ;
     60                                ;  Check for storing activation character.
     61                                ;
     62 016064  032705  000000G                    BIT     #ACFLAG,R5      ;IS THIS AN ACTIVATION CHAR?
     63 016070  001414                             BEQ     3$              ;BRANCH IF NOT
     64 016072  005261  000000G                    INC     LACTIV(R1)      ;Count another pending activ char
     65 016076  016161  000000G 000000G            MOV     LINNXT(R1),LSTACT(R1);REMEMBER POS OF CHAR
     66 016104  032761  000000G 000000G            BIT     #$DODFR,LSW3(R1);DOING DEFERED ECHOING?
     67 016112  001003                             BNE     3$              ;BRANCH IF YES
     68 016114  042761  000000G 000000G            BIC     #$1STCH,LSW3(R1);SAY NO CHARS AFTER ACTIVATION
     69                                ;
     70                                ;  See if we need to trigger a completion routine for TT input
     71                                ;
     72 016122  005761  000000G        3$:     TST     LTTCR(R1)       ;Does job want TT input compl routine?
     73 016126  001403                         BEQ     10$             ;Br if not
     74 016130                                 OCALL   TTCPL           ;Trigger completion routine
     75                                ;
     76                                ;  Finished
     77                                ;
     78 016136  000241                 10$:    CLC                     ;SIGNAL NO BUFFER OVERFLOW
     79 016140  012603                 6$:     MOV     (SP)+,R3
     80 016142  012602                         MOV     (SP)+,R2
     81 016144  012600                         MOV     (SP)+,R0
     82 016146  000207                         RETURN
     83                                ;
     84                                ;  Signal buffer overflow
     85                                ;
     86 016150  052761  000000G 000000G 5$:    BIS     #$TTERR,LSW4(R1);REMEMBER THAT AN ERROR OCCURED
     87 016156  012700  000000G               MOV     #BELL,R0        ;Get a bell character
     88 016162  004737  017072'               CALL    ECHO2           ;Ring the bell to signal buffer nearly full
     89 016166  000261                         SEC                     ;SIGNAL OVERFLOW
     90 016170  000763                         BR      6$
```

```
                                        .SBTTL  STRACT -- Store activation character
                                ;-----------------------------------------------------------------------
                                ;  STRACT is called to store an activation character
                                ;  which is contained in R5.  The high order bit of
                                ;  the word (ACFLAG) is set on to indicate to GETCHR
                                ;  that the character is an activation character.
                                ;  After the character is stored the user is activated.
                                ;  When called, R5 must contain the character to be stored.
                                ;  R1 must contain the user index #.
                                ;  All registers are preserved.
                                ;
016172  052705  0000000         STRACT: BIS     #ACFLAG,R5      ;SET ACTIVATION FLAG WITH CHAR
016176  004737  015674'                 CALL    STRCHR          ;STORE THE CHARACTER
016202  042705  0000000                 BIC     #ACFLAG,R5      ;RESET THE FLAG
016206  032761  0000000 0000000         BIT     #$DBGMD,LSW6(R1);IS DEBUGGER DOING I/O NOW?
016214  001004                          BNE     3$              ;BR IF YES - DON'T RESET ACTIVATION INFO
016216  005061  0000000                 CLR     LAFSIZ(R1)      ;RESET FIELD-WIDTH ACTIVATION
016222  005061  0000000                 CLR     LFWLIM(R1)      ;CLEAR FIELD WIDTH LIMIT
                                ;
                                ;  See if we should begin deferred char echoing.
                                ;
016226  032761  0000000 0000000 3$:     BIT     #$DEFER,LSW2(R1);DEFERRED MODE WANTED?
016234  001403                          BEQ     1$              ;BRANCH IF NOT
016236  052761  0000000 0000000         BIS     #$DODFR,LSW3(R1);BEGIN DEFERRED ECHOING
                                ;
                                ;  Activate the job
                                ;
016244  004737  015472'         1$:     CALL    INFIN           ;ACTIVATE THE USER
016250  000207                          RETURN
```

```
      1                                         .SBTTL  STRSNG -- Store char with single-character input
      2                                 ;-----------------------------------------------------------------
      3                                 ;  STRSNG is called to store a character received while in
      4                                 ;  single character activation mode.
      5                                 ;
      6                                 ;  Inputs:
      7                                 ;   R1 = Job index number.
      8                                 ;   R5 = Character received.
      9                                 ;
     10 016252  010546                  STRSNG: MOV     R5,-(SP)        ;Save input character
     11                                 ;
     12                                 ;  Store the activation character
     13                                 ;
     14 016254  052705  0000000                 BIS     #ACFLAG,R5      ;Set activation flag with char
     15 016260  004737  015674'                 CALL    STRCHR          ;Store the character
     16                                 ;
     17                                 ;  See if we need to begin deferred echo mode
     18                                 ;
     19 016264  032761  0000000 0000000         BIT     #$DEFER,LSW2(R1);Deferred mode wanted?
     20 016272  001403                          BEQ     4$              ;Branch if not
     21 016274  052761  0000000 0000000         BIS     #$DODFR,LSW3(R1);Begin deferred echoing
     22                                 ;
     23                                 ;  Activate the job
     24                                 ;
     25 016302  004737  015472'         4$:     CALL    INFIN           ;Activate the job
     26                                 ;
     27                                 ;  Finished
     28                                 ;
     29 016306  012605                  9$:     MOV     (SP)+,R5        ;Recover the original character
     30 016310  000207                          RETURN
```

```
                                        .SBTTL  FETCHR -- Fetch next char from TT input ring buffer
                                ;-----------------------------------------------------------------------
                                ;  FETCHR is called to fetch the next character from the TT input
                                ;  ring buffer.
                                ;
                                ;  Inputs:
                                ;   R1 = Line index number.
                                ;   R2 = Pointer to character in TT input ring buffer.
                                ;
                                ;  Outputs:
                                ;   C-flag cleared ==> A char was gotten.
                                ;   C-flag set     ==> No more chars in TT input ring buffer.
                                ;   R0 = Character gotten (if C-flag cleared).
                                ;        The ACFLAG flag is set in R0 if the char is an activation char.
                                ;   R2 = Updated to point to next character.
                                ;
016312  010446                  FETCHR: MOV     R4,-(SP)
016314  010546                          MOV     R5,-(SP)
                                ;
                                ;  See if there is another character in the ring buffer
                                ;
016316  020261  0000000                 CMP     R2,LINNXT(R1)   ;Are there any more chars in the buffer?
016322  001450                          BEQ     8$              ;Br if not
                                ;
                                ;  Compute index into bit vector that indicates if character is an
                                ;  activation character.
                                ;
016324  010204                          MOV     R2,R4           ;Get pointer to character
016326  166104  0000000                 SUB     LINBUF(R1),R4   ;Get character index into buffer
016332  073427  177775                  ASHC    #-3,R4          ;Get byte index into vector in R2
016336  072527  177763                  ASH     #-13.,R5        ;Right justify bit-within-byte index
016342  042705  177770                  BIC     #^C7,R5         ;Clear possible sign extension from shift
016346  066104  0000000                 ADD     LINEND(R1),R4   ;Get address of byte with flag bit
                                ;
                                ;  Get next character from TT ring buffer.
                                ;
016352                                  DISABL                  ;;;** Disable interrupts **
016360                                  TTMAP                   ;;;Map to TT buffer area
016374  005000                          CLR     R0              ;;;Initially clear R0
016376  152200                          BISB    (R2)+,R0        ;;;Get character without sign extension
016400  136514  000340'                 BITB    BITMSK(R5),(R4) ;;;Is this an activation character?
016404  001402                          BEQ     1$              ;;;Br if not
016406  052700  0000000                 BIS     #ACFLAG,R0      ;;;Remember this is an activation character
016412                          1$:     UNMAP                   ;;;Restore mapping
016420                                  ENABL                   ;** Enable interrupts **
                                ;
                                ;  See if we need to wrap around to front of ring buffer
                                ;
016426  020261  0000000         3$:     CMP     R2,LINEND(R1)   ;Did we just go past end of ring buffer?
016432  103402                          BLO     4$              ;Br if not
016434  016102  0000000                 MOV     LINBUF(R1),R2   ;Wrap around to front of ring buffer
                                ;
                                ;  We got a character
                                ;
016440  000241                  4$:     CLC                     ;Signal that we got a character
016442  000401                          BR      9$
                                ;
```

```
                                ;  There are no more characters in the buffer
                                ;
016444  000261                8$:     SEC                     ;Signal that there are no more characters
                                ;
                                ;  Finished
                                ;
016446  012605                9$:     MOV     (SP)+,R5
016450  012604                        MOV     (SP)+,R4
016452  000207                        RETURN
```

```
                                        .SBTTL  INSCHR -- Insert character into TT input ring buffer
                                ;-----------------------------------------------------------------------
                                ;  This routine is called to insert a character into a specified position
                                ;  in the TT input-character buffer for a line.  If there is an existing
                                ;  character in the position, it is overwritten.
                                ;
                                ;  Inputs:
                                ;   R0 = Character to be stored (optionally with ACFLAG flag).
                                ;   R1 = Line index number.
                                ;   R2 = Pointer to position in buffer where char is to be stored
                                ;
                                ;  Outputs:
                                ;   R2 = Pointer to next character position in buffer.
                                ;
016454  010446                  INSCHR: MOV     R4,-(SP)
016456  010546                          MOV     R5,-(SP)
                                ;
                                ;  Compute pointer into parallel bit vector with activation-character flags
                                ;
016460  010204                          MOV     R2,R4           ;Get character buffer pointer
016462  166104  0000000                 SUB     LINBUF(R1),R4   ;Compute byte index into buffer
016466  073427  177775                  ASHC    #-3,R4          ;Get byte index in R4
016472  072527  177763                  ASH     #-13.,R5        ;Right justify bit-within-byte index
016476  042705  177770                  BIC     #^C7,R5         ;Clear possible sign extension
016502  116505  000340'                 MOVB    BITMSK(R5),R5   ;Get bit mask
016506  066104  0000000                 ADD     LINEND(R1),R4   ;Get address of byte with activation flag
                                ;
                                ;  Store character into buffer and set or clear the activation-character flag
                                ;
016512                                  DISABL                  ;;;** Disable interrupts **
016520                                  TTMAP                   ;;;Map to TT buffer area
016534  110022                          MOVB    R0,(R2)+        ;;;Store character into buffer
016536  140514                          BICB    R5,(R4)         ;;;Clear the activation-character flag
016540  032700  0000000                 BIT     #ACFLAG,R0      ;;;Is this an activation character?
016544  001401                          BEQ     2$              ;;;Br if not
016546  150514                          BISB    R5,(R4)         ;;;Set the activation-character flag
016550                          2$:     UNMAP                   ;;;Restore mapping
016556                                  ENABL                   ;** Enable interrupts **
                                ;
                                ;  See if we need to wrap around buffer pointer
                                ;
016564  020261  0000000                 CMP     R2,LINEND(R1)   ;Do we need to wrap around to buffer front?
016570  103402                          BLO     9$              ;Br if not
016572  016102  0000000                 MOV     LINBUF(R1),R2   ;Wrap around to front of buffer
                                ;
                                ;  Finished
                                ;
016576  012605                  9$:     MOV     (SP)+,R5
016600  012604                          MOV     (SP)+,R4
016602  000207                          RETURN
```

```
                                        .SBTTL  DELCHR -- Delete character from TT input ring buffer
                                ;-----------------------------------------------------------------------
                                ;  DELCHR is called to remove a character from the TT input ring buffer.
                                ;  If there are other characters in the ring buffer in front of the one
                                ;  being deleted, the ring buffer is compressed.
                                ;
                                ;  Inputs:
                                ;   R1 = Line index number.
                                ;   R2 = Pointer to character to delete.
                                ;
                                ;  Outputs:
                                ;   C-flag set ==> No more characters in buffer.
                                ;   R0 = Deleted character (Same format as FETCHR).
                                ;   R2 = Pointer to character that follows deleted character.
                                ;
016604  010346                  DELCHR: MOV     R3,-(SP)
016606  010546                          MOV     R5,-(SP)
                                ;
                                ;  Get the character being deleted
                                ;
016610  010203                          MOV     R2,R3           ;Save original character pointer
016612  004737  016312'                 CALL    FETCHR          ;Get character being deleted
016616  103502                          BCS     20$             ;Br if no character to delete
                                ;
                                ;  See if we are deleting the 1st character in the buffer
                                ;
016620  020361  0000000                 CMP     R3,LINPNT(R1)   ;Is this the 1st char in the buffer?
016624  001003                          BNE     3$              ;Br if not
016626  010261  0000000                 MOV     R2,LINPNT(R1)   ;Set new pointer to 1st char in buffer
016632  000434                          BR      8$              ;Go update buffer counts
                                ;
                                ;  We are not deleting the 1st character in the buffer.
                                ;  If we are deleting a character from the middle of the buffer, we move
                                ;  over any following characters to compress the free space.
                                ;  If we are deleting the last character in the buffer, no compression
                                ;  is necessary.
                                ;
016634  010046                  3$:     MOV     R0,-(SP)        ;Save character that is being deleted
016636  010346                          MOV     R3,-(SP)        ;Save pointer to deleted char position
016640                          4$:     DISABL                  ;;;** Disable interrupts **
016646  020261  0000000                 CMP     R2,LINNXT(R1)   ;;;Are we at the end of the chars now?
016652  001415                          BEQ     5$              ;;;Br if yes
016654                                  ENABL                   ;** Enable interrupts **
                                ;
                                ;  Slide characters over in buffer to fill in gap left by deleted character
                                ;
016662  010205                          MOV     R2,R5           ;Save pointer to following character
016664  004737  016312'                 CALL    FETCHR          ;Get next character in buffer
016670  010246                          MOV     R2,-(SP)        ;Save updated pointer
016672  010302                          MOV     R3,R2           ;Get pointer to pos to store character
016674  004737  016454'                 CALL    INSCHR          ;Reinsert the character
016700  010503                          MOV     R5,R3           ;Advance insert pointer
016702  012602                          MOV     (SP)+,R2        ;Get pointer to next char to move over
016704  000755                          BR      4$              ;Loop until all characters have been moved
                                ;
                                ;  We have moved over all characters that followed the deleted one.
                                ;  Save new pointer to the end of the characters in the buffer.
```

```
                                ;
016706  010361  000000G          5$:     MOV     R3,LINNXT(R1)   ;;;Set new pointer to end of chars in buffer
016712                                  ENABL                   ;** Enable interrupts **
016720  012602                          MOV     (SP)+,R2        ;Get pointer to char following deleted one
016722  012600                          MOV     (SP)+,R0        ;Get character being deleted
                                ;
                                ;  Update character counters
                                ;
016724  005261  000000G          8$:     INC     LINSPC(R1)      ;Another free char space in buffer
016730  005361  000000G                  DEC     LINCNT(R1)      ;One less char in buffer
016734  032700  000000G                  BIT     #ACFLAG,R0      ;Is deleted char an activation char?
016740  001402                          BEQ     19$             ;Br if not
016742  005361  000000G                  DEC     LACTIV(R1)      ;One fewer pending activation chars
                                ;
                                ;  If we sent an XOFF to the terminal,
                                ;  send an XON if the input buffer is nearly empty.
                                ;
016746  032761  000000G 000000G 19$:     BIT     #$XSTOP,LSW6(R1);Have we stopped input from silo buffer?
016754  001422                          BEQ     16$             ;Br if not
016756  026127  000000G 000017          CMP     LINCNT(R1),#15. ;Is input buffer almost empty?
016764  101406                          BLOS    17$             ;Br if yes
016766  032700  000000G                  BIT     #ACFLAG,R0      ;Is this an activation character?
016772  001413                          BEQ     16$             ;Br if not
016774  005761  000000G                  TST     LACTIV(R1)      ;Is this last activation char in buffer?
017000  001010                          BNE     16$             ;Br if not
017002  042761  000000G 000000G 17$:     BIC     #$XSTOP,LSW6(R1);Reenable input from silo
017010  052761  000000G 000000G          BIS     #$NDICP,LSW10(R1);Say line needs input character servicing
017016  005237  000000G                  INC     NEDCDI          ;Say input character processing needed
                                ;
                                ;  We deleted a character
                                ;
017022  000241                  16$:     CLC                     ;Signal that we deleted a character
                                ;
                                ;  Finished
                                ;
017024  012605                  20$:     MOV     (SP)+,R5
017026  012603                          MOV     (SP)+,R3
017030  000207                          RETURN
```

```
                                        .SBTTL  ECHO   -- Echo character to terminal
                                ;-------------------------------------------------------
                                ;  Subroutine ECHO is called to echo the character in R0.
                                ;  User index must be in R1.
                                ;  ECHO1 does not check for rubout character echoing.
                                ;  ECHO2 does not check for deferred character echoing.
                                ;  All registers are preserved.
                                ;
017032  004737  017160'         ECHO:   CALL    RBEND           ;TERMINATE ANY RUBOUT FIELD
017036  004737  017414'         ECHO1:  CALL    CVTLC           ;CONVERT LOWER CASE CHARS TO UPPER CASE
017042  032761  0000000 0000000        BIT     #$DODFR,LSW3(R1);ARE WE DEFERRING ECHO NOW?
017050  001016                         BNE     ECHOR           ;BRANCH IF WE ARE
017052  032761  0000000 0000000        BIT     #$ECHO,LSW2(R1) ;IS CHARACTER ECHOING WANTED?
017060  001004                         BNE     ECHO2           ;BR IF YES
017062  032761  0000000 0000000        BIT     #$DBGMD,LSW6(R1);IS A DEBUGGER USING TERMINAL NOW?
017070  001406                         BEQ     ECHOR           ;BR IF NOT
017072  026127  0000000 000017 ECHO2:  CMP     LOTSPC(R1),#15. ;ROOM TO ECHO CHAR?
017100  002402                         BLT     ECHOR           ;BRANCH IF NOT
017102  004737  003122'               CALL    PUTCH2          ;PUT CHAR IN OUTPUT BUFFER
017106  000207                 ECHOR:  RETURN

                                        .SBTTL  ECOCTL -- Echo a control character
                                ;-------------------------------------------------------
                                ;  ECOCTL is called to echo certain control characters
                                ;  such as ctrl-C and ctrl-U.  When called R5 must
                                ;  contain the control character.  The control character
                                ;  is converted to a printing ascii char and printed
                                ;  following an up arrow and before a cr-lf.
                                ;  When called R1 must contain the line index.
                                ;  all registers are preserved.
                                ;
017110  010046                 ECOCTL: MOV     R0,-(SP)
017112  112700  000136                 MOVB    #136,R0         ;ECHO '^'
017116  004737  017032'                CALL    ECHO
017122  010500                         MOV     R5,R0           ;GET CONTROL CHARACTER
017124  052700  000100                 BIS     #100,R0         ;CONVERT TO PRINTING CHAR
017130  004737  017036'                CALL    ECHO1           ;PRINT CONTROL CHAR
017134  112700  0000000                MOVB    #CR,R0          ;PRINT CR-LF
017140  004737  017036'                CALL    ECHO1
017144  112700  0000000                MOVB    #LF,R0
017150  004737  017036'                CALL    ECHO1
017154  012600                         MOV     (SP)+,R0
017156  000207                         RETURN

                                        .SBTTL  RBEND  -- Terminate rubout sequence
                                ;-------------------------------------------------------
                                ;  RBEND is called to terminate any current rubout field.
                                ;  If a rubout field is in progress RBEND puts out
                                ;  a back slash and terminates the rubout state.
                                ;  When called R1 must contain the user index #.
                                ;  All registers are preserved.
                                ;
017160  032761  0000000 0000000 RBEND:  BIT     #$RBOUT,LSW3(R1);ARE WE IN A RUBOUT FIELD?
017166  001411                         BEQ     1$              ;BRANCH IF NOT
017170  042761  0000000 0000000        BIC     #$RBOUT,LSW3(R1);CLEAR RUBOUT STATE
017176  010046                         MOV     R0,-(SP)
017200  112700  000134                 MOVB    #'\,R0          ;OUTPUT BACK SLASH
```

```
     58 017204  004737  017036'                    CALL     ECHO1
     59 017210  012600                             MOV      (SP)+,R0
     60 017212  000207                       1$:   RETURN
```

```
                                        .SBTTL  SCACHK -- Check for single-character activation
                                ;-----------------------------------------------------------------------
                                ;  SCACHK is called to determine if this job is in single character
                                ;  activation mode.
                                ;
                                ;  Inputs:
                                ;   R1 = Job index number.
                                ;
                                ;  Outputs:
                                ;   C-flag set if in single character activation mode.
                                ;
017214  032761  0000000 0000000 SCACHK: BIT     #SPCTTY,LJSW(R1);Does program want single char input?
017222  001412                          BEQ     1$              ;Br if not
017224  032761  0000000 0000000         BIT     #$CHACT,LSW5(R1);Is single character activation enabled?
017232  001406                          BEQ     1$              ;Br if not
017234  032761  0000000 0000000         BIT     #$DBGMD,LSW6(R1);Is a debugger using terminal now?
017242  001002                          BNE     1$              ;Br if yes
                                ;
                                ;  Job is in single character activation mode
                                ;
017244  000261                          SEC                     ;Signal that single char activation wanted
017246  000401                          BR      9$
                                ;
                                ;  Job does not want single char activation
                                ;
017250  000241                  1$:     CLC                     ;Signal no single char activation
                                ;
                                ;  Finished
                                ;
017252  000207                  9$:     RETURN
```

```
                                        .SBTTL  SLCHK  -- Check for single line editor mode
                                ;-----------------------------------------------------------------------
                                ;  SLCHK is called to determine if this job is in either single character
                                ;  activation mode or if the input is going to the single line editor.
                                ;
                                ;  Inputs:
                                ;   R1 = Job index number
                                ;
                                ;  Outputs:
                                ;   C-flag set if in single-character-activation or SL mode.
                                ;   All registers are preserved.
                                ;
017254  032761  0000000 0000000 SLCHK:  BIT     #SPCTTY,LJSW(R1);Does program want single char input?
017262  001411                          BEQ     1$              ;Br if not
017264  032761  0000000 0000000         BIT     #$CHACT,LSW5(R1);Is single character activation enabled?
017272  001437                          BEQ     2$              ;Br if not
017274  032761  0000000 0000000         BIT     #$DBGMD,LSW6(R1);Is a debugger using terminal now?
017302  001033                          BNE     2$              ;Br if yes
017304  000430                          BR      3$              ;We are in single character mode
                                ;
                                ;  We are not in single character activation mode.
                                ;  See if we are in Single Line Editor mode.
                                ;
017306  032761  0000000 0000000 1$:     BIT     #$SLON,LSW7(R1) ;Is SL enabled for this line?
017314  001426                          BEQ     2$              ;Br if not
017316  032761  0000000 0000000         BIT     #DISSLE,LJSW(R1);Did program disable SL?
017324  001022                          BNE     2$              ;Br if yes
017326  032761  0000000 0000000         BIT     #<$ODTMD!$HITTY>,LSW4(R1);Are we in ODT or high efficiency?
017334  001016                          BNE     2$              ;Br if yes
017336  032761  0000000 0000000         BIT     #$VTESC,LSW5(R1);VTxxx activation enabled?
017344  001012                          BNE     2$              ;Br if yes
017346  032761  0000000 0000000         BIT     #$GTLIN,LSW4(R1);Is a .GTLIN being done?
017354  001004                          BNE     3$              ;Br if yes
017356  032761  0000000 0000000         BIT     #$SLTTY,LSW7(R1);Is SL enabled for TTY input?
017364  001402                          BEQ     2$              ;Br if not
                                ;
                                ;  We are in single line activation or single line editor mode.
                                ;
017366  000261                  3$:     SEC                     ;Signal single character mode
017370  000401                          BR      9$
                                ;
                                ;  We are not in single character mode.
                                ;
017372  000241                  2$:     CLC                     ;Signal not single character mode
                                ;
                                ;  Finished
                                ;
017374  000207                  9$:     RETURN
```

```
                                        .SBTTL  SCACHR -- Handle single-character activation characters
                                ;-----------------------------------------------------------------------
                                ;  SCACHR is called to check to see if the program is in single-character
                                ;  activation mode.  If yes then the current character is passed to the
                                ;  program as an activation character.
                                ;  If not then the carry-flag is cleared on return.
                                ;
                                ;  Inputs:
                                ;   R1 = Virtual line index number.
                                ;   R5 = Current input character.
                                ;
                                ;  Outputs:
                                ;   C-flag set     ==> In single-char activation mode. Char processed.
                                ;   C-flag cleared ==> Not in single character activation mode.
                                ;
017376  004737  017254'         SCACHR: CALL    SLCHK           ;Are we in single-char activation mode?
017402  103003                         BCC     9$              ;Br if not
                                ;
                                ;  We are in single-character activation mode.
                                ;  Pass the character to the program as an activation char.
                                ;
017404  004737  016252'                 CALL    STRSNG          ;Store character and activate
017410  000261                         SEC                     ;Say character was processed by us
                                ;
                                ;  Finished
                                ;
017412  000207                 9$:     RETURN
```

```
                                        .SBTTL  CVTLC  -- Convert lower-case chars to upper-case
                                ;-----------------------------------------------------------------
                                ;  CVTLC is called to see if TT input characters entered
                                ;  in lower case should be translated to upper case.
                                ;  When called, R0 must contain the character to be tested.
                                ;  R1 must contain the user line index #.
                                ;  CVTLC checks to see if set-lc has been done and if the
                                ;  bit is set in the JSW allowing lower case characters.
                                ;  On return, the resulting character is returned in R0.
                                ;  All other registers are preserved.
                                ;
017414  032761  000000@ 000000@ CVTLC:  BIT     #$LC,LSW2(R1)   ;WAS "SET TT LC" DONE?
017422  001404                          BEQ     1$              ;BR IF NOT
017424  032761  000000@ 000000@         BIT     #LCBIT,LJSW(R1) ;IS LC-BIT SET IN JSW?
017432  001014                          BNE     2$              ;BR IF YES (LC OK)
                                ;
                                ;  Translate lower case to upper case
                                ;
017434  010046                  1$:     MOV     R0,-(SP)        ;SAVE ORIGINAL CHARACTER VALUE
017436  042700  000000@                 BIC     #ACFLAG,R0      ;CLEAR ACTIVATION FLAG
017442  020027  000141                  CMP     R0,#141         ;LC('A')
017446  002405                          BLT     3$              ;BR IF NOT LOWER-CASE LETTER
017450  020027  000172                  CMP     R0,#172         ;LC('Z')
017454  101002                          BHI     3$              ;BR IF NOT LOWER-CASE LETTER
017456  042716  000040                  BIC     #40,(SP)        ;CONVERT LOWER-CASE LETTER TO UPPER-CASE
017462  012600                  3$:     MOV     (SP)+,R0        ;GET POSSIBLY CONVERTED LETTER BACK TO R0
017464  000207                  2$:     RETURN
```

```
                                        .SBTTL  SIGWAT -- Signal virtual line wait condition
                                ;-----------------------------------------------
                                ;  SIGWAT IS CALLED TO SIGNAL THE USER THAT ONE OF HIS
                                ;  VIRTUAL LINES IS ENTERING A WAIT CONDITION.
                                ;  IF THE LINE ENTERING THE WAIT STATE IS NOT THE ONE
                                ;  WHICH IS CURRENTLY ASSOCIATED WITH THE TERMINAL A BELL
                                ;  IS SENT TO THE USER'S TERMINAL.
                                ;
                                ;  Inputs:
                                ;   R1 = Job index of job entering wait condition.
                                ;   All registers are preserved.
                                ;
017466  010046                 SIGWAT: MOV     R0,-(SP)
017470  010146                         MOV     R1,-(SP)
                                ;
                                ;  Only signal if the job that is entering the wait state is not
                                ;  currently connected to the terminal.
                                ;
017472  016100  000000                  MOV     LNPRIM(R1),R0   ;GET PRIMARY LINE #
017476  026001  000000                  CMP     LNMAP(R0),R1    ;IS THE LINE CONNECTED TO TERM?
017502  001415                          BEQ     9$              ;IF YES THEN NO NEED TO SIGNAL
                                ;
                                ;  If we have already signaled that job is in a wait state, don't
                                ;  signal again until user reconnects to this job.
                                ;
017504  032761  000000 000000           BIT     #$VBELL,LSW9(R1);Have we already signaled wait state?
017512  001011                          BNE     9$              ;Br if yes
017514  052761  000000 000000           BIS     #$VBELL,LSW9(R1);Set flag saying we have signaled
                                ;
                                ;  Send bell to signal wait condition
                                ;
017522  016001  000000                  MOV     LNMAP(R0),R1    ;GET CURRENTLY CONNECTED LINE #
017526  112700  000000                  MOVB    #BELL,R0        ;SEND BELL AS SIGNAL CHARACTER
017532  004737  005722'                 CALL    TRYCHR
                                ;
                                ;  Finished
                                ;
017536  012601                  9$:     MOV     (SP)+,R1
017540  012600                          MOV     (SP)+,R0
017542  000207                          RETURN

                                        .SBTTL  SIGBRK -- Signal program that Break character was received
                                ;-----------------------------------------------------------------------
                                ;  SIGBRK is called to signal a program that a Break character was received.
                                ;  If the program has requested notification of Break character reception,
                                ;  an asynchronous completion routine request is queued for the program.
                                ;
                                ;  Inputs:
                                ;   R1 = Virtual line number.
                                ;
017544  010446                  SIGBRK: MOV     R4,-(SP)
017546  016104  000000                  MOV     LBRKCQ(R1),R4   ;DOES USER WANT NOTIFICATION OF BREAK?
017552  001404                          BEQ     1$              ;BR IF NOT
017554  005061  000000                  CLR     LBRKCQ(R1)      ;SAY THAT BREAK QUEUE ELEMENT HAS BEEN USED UP
017560  004737  000000                  CALL    QCOMPL          ;QUEUE COMPLETION ROUTINE FOR THE JOB
017564  012604                  1$:     MOV     (SP)+,R4
017566  000207                          RETURN
```

```
     58          000000                                .END
Errors detected:   0

*** Assembler statistics


Work  file  reads:  0
Work  file writes:  0
Size of work file:  10440 Words  ( 41 Pages)
Size of core pool:  17920 Words  ( 70 Pages)
Operating  system:  RT-11

Elapsed time:  00:02:06.50
DK:TSTTY,LP:TSTTY=DK:TSTTY.MAC/C/N:SYM
```

Cross reference table (CREF V05.04)

$1CTLC	1-55	51-21	53-44	54-40	55-14	55-19	57-29	60-14	61-29	62-16	63-40	64-14
$1ESC	1-43	50-77	51-20	53-45	54-41	55-20	57-30	60-15	61-30	62-17	63-30	63-33
	63-36	64-15	65-21	65-48	65-62							
$1STCH	1-48	47-25	55-70	60-40	70-15	70-17	70-68					
$8BIT	1-52	12-31										
$AUTO	1-34	49-32										
$CCLRN	1-76	35-38										
$CFABT	1-86	9-120	9-122									
$CFALL	1-51	39-75	39-82	41-14	43-35							
$CFCCL	1-87	9-95	43-12									
$CFDCC	1-87	9-30	9-97									
$CFKIL	1-87	55-66										
$CFOPN	1-51	43-22	43-25	43-123								
$CFSOT	1-54	13-27	39-54	39-61	39-68	43-35						
$CHACT	1-44	28-23	28-30	33-75	34-11	57-11	61-11	63-13	77-14	78-15		
$CTRLC	1-43	12-22	19-17	33-14	55-65							
$CTRLD	1-40	56-15										
$CTRLO	1-48	9-113	13-34	18-32	58-9	58-17	58-25					
$CTRLS	1-49	55-63										
$CTRLW	1-48	50-75	50-78	50-83	50-85	50-89						
$DBGBK	1-56	56-17										
$DBGMD	1-82	7-22	35-32	38-25	50-28	51-49	51-63	53-34	55-40	60-92	64-26	64-93
	71-15	76-15	77-16	78-17								
$DBKMN	1-40	56-12										
$DEAD	1-57	49-27										
$DEBUG	1-56	56-15										
$DEFER	1-46	25-49	25-56	71-22	72-19							
$DETCH	1-43	13-21	19-15	19-17	24-13	26-10	27-13	28-5	28-14	29-13	33-34	46-7
$DILUP	1-42	49-19										
$DISCN	1-42	12-22	19-17	33-14	33-36	46-9						
$DODFR	1-49	25-57	35-12	47-16	47-54	55-61	60-25	70-15	70-66	71-24	72-21	76-11
$DOOFF	1-42	7-19	33-29									
$ECHO	1-44	25-5	25-12	35-48	36-18	38-10	64-24	76-13				
$FLAGC	1-54	7-26	7-28	7-34								
$FORM	1-45	16-46										
$FORMO	1-89	2-18										
$GCECO	1-49	25-57	35-14	35-16	36-6	47-24	55-61					
$GCESC	1-49	33-13	33-95	33-102	38-12							
$GTLIN	1-51	9-24	9-148	33-49	41-12	78-32						
$HITTY	1-54	6-37	8-9	8-20	12-16	28-16	33-45	33-85	50-57	78-28		
$INKMN	1-50	9-56	9-71	9-135	43-42	43-49	43-140	56-10				
$LC	1-46	25-31	25-42	80-12								
$LOFCF	1-59	7-17	33-27	43-36	51-15	55-30						
$NDICP	1-43	46-18	75-84									
$NOIN	1-47	7-21	12-24	43-38	46-29	55-34						
$NOINT	1-45	33-58	46-35									
$NOLF	1-81	30-19	30-26	38-23	53-36							
$NOOUT	1-46	12-38	20-74	20-81	34-41							
$NOVLN	1-46	25-24										
$NOWIN	1-37	17-17	43-52	43-102								
$NOWTT	1-39	6-17	7-24	28-37								
$NTGCC	1-41	9-39	9-43	9-99								
$ODTMD	1-51	33-45	51-32	53-22	54-24	66-12	78-28					
$PWKEY	1-37	52-16										
$QUIET	1-50	9-104	35-21	35-46	36-16	39-53	39-61	39-68				
$RBOUT	1-48	60-24	64-34	64-38	76-53	76-55						

Cross reference table (CREF V05.04)

```
$RBRK     1-34     49-53    50-15    50-18
$RFRSH    1-34     55-36
$RTCS     1-36     55-21    65-22    65-31    65-36    65-58    65-67
$SCCA     1-34     43-51    43-96    43-99    55-32
$SCOPE    1-44     60-30    64-28
$SETCC    1-50     55-45
$SLINI    1-63     55-60
$SLON     1-38     33-43    78-24
$SLTTY    1-38     33-51    78-34
$SUCF     1-59     43-39    46-30    55-30
$SUSPN    1-35     55-67
$TAB      1-45     16-29
$TAPE     1-86     30-5     30-12    54-13
$TRNSP    1-47     12-50    25-64
$TTERR    1-54      8-59     8-61    70-86
$UCTLC    1-50     24-24    26-35    35-34
$UKMON    1-58      9-58
$V52EM    1-38     60-71
$VBELL    1-35     81-26    81-28
$VTESC    1-55     23-17    23-24    29-38    33-47    33-100   47-40    50-64    78-30
$XSTOP    1-86     46-15    46-17    48-21    70-27    75-75    75-83
...V1    40-37     40-37    43-24    43-121   43-121   43-127   43-127   45-28    45-28
...V2    40-37     40-37    40-37#   40-37#   43-24    43-24#   43-121   43-121   43-121#  43-121#  43-127   43-127
        43-127#   43-127#   45-28    45-28    45-28#   45-28#
ABORT     1-94     45-33
ACFLAG    1-69     24-43    24-45    27-39    27-42    29-34    33-78    35-6     38-17    47-47    68-34    70-34
         70-62     71-12    71-14    72-14    73-43    74-34    75-68    75-79    80-20
AF$CCA    1-34     43-97
AF$NPW    1-35     43-100
AFCF      1-34     43-48*   43-95*   43-97    43-100
ASKLIN    1-25      9-12#
AUTSPD    1-36     49-34
BELL      1-67     29-50    51-65    70-87    81-33
BITMSK    1-175#   73-41    74-25
BKSPAC    1-58      1-66    60-63    60-95    60-99    64-64    64-74    64-81    64-90    64-97
BRKPT     1-87     59-10
BUFCHR    1-21      1-26    17-24    18-12#
CCVECT   21-35     22-4#    22-30
CFACFL    1-34     42-15    42-39
CFARG     1-40     40-37    43-121   43-127   45-28
CFBLK     1-76     40-36*   40-37    43-117*  43-127
CFBUF     1-63     40-37    40-49    43-127
CFCHAN    1-63     40-37    40-37    43-24    43-24    43-121   43-121   43-127   43-127
CFCHAR   39-26     39-34    40-15#
CFEND     1-63     40-32
CFHOLD    1-44      9-103*  39-19    39-21*   39-116*  43-18*
CFIND     1-91     43-17    43-83*
CFLAG     1-89      3-17
CFLFL4    1-92     43-87
CFNEST    1-91     43-29    43-130*
CFPNT     1-61      9-100    9-111    9-118    13-29    25-71    40-31    40-45    40-52*   41-10    42-10*   42-33*
         43-13
CFPOP    40-44     43-6#
CFSP      1-91     43-60    43-131*
CFSPND    1-69      9-48*    9-100*   9-144    9-147*   25-73*   25-81    25-84*
CFSTOP    9-101    25-74     42-6#    43-34
```

Cross reference table (CREF V05.04)

```
CFSTRT    9-146    25-83    42-29#   43-116
CFTEST    7-15     27-16    29-18    35-4     35-40    35-44    36-14    39-14    41-10#
CFTNO    41-11     41-15    41-18#
CFTST1   41-12#
CHKABT    1-73      2-35     2-38     2-56     2-59     4-19     4-22    11-28    11-31    31-15    39-97    46-31
CHKODT   51-34     66-12#
CHNADR    1-90      2-15*    3-14     3-16*    3-33*
CKCW     50-29     50-65    50-67    50-73#
CKHIIN   50-50     50-57#
CKICTL   50-86     50-88    50-99#
CKSPAC   50-34     50-36    50-42#
CKVTAC   50-68     65-14#
CKVTES   50-43     50-58    50-64#
CMDA     22-4      23-5#
CMDB      1-28     22-5     23-17#
CMDC      1-28     22-6     23-24#
CMDD     22-7      24-5#
CMDE      1-28     22-8     25-5#
CMDF      1-28     22-9     25-12#
CMDG      1-28     22-10    25-18#
CMDH      1-28     22-11    25-24#
CMDI      1-28     22-12    25-31#
CMDJ      1-29     22-13    25-42#
CMDK      1-29     22-14    25-49#
CMDL      1-29     22-15    25-56#
CMDM      1-29     22-16    25-64#
CMDN      1-29     22-17    25-71#
CMDO      1-29     22-18    25-81#
CMDP     22-19     26-5#
CMDQ     22-20     27-5#
CMDR      1-29     22-21    28-5#
CMDS      1-30     22-22    28-23#
CMDT      1-30     22-23    28-30#
CMDU      1-30     22-24    28-37#
CMDV     22-25     29-5#
CMDW      1-30     22-26    30-5#
CMDX      1-30     22-27    30-12#
CMDY      1-30     22-28    30-19#
CMDZ      1-30     22-29    30-26#
CORUSR    1-61      9-23    10-10    11-15    33-12    43-11
CR        1-58      9-128   20-44    24-55    29-32    37-10    38-58    44-34    49-37    54-26    54-35    60-57
         70-32     76-38
CS$EOF    1-89      2-15     3-14     3-16     3-33
CSEMFO    1-155     1-166#
CSEMID    1-153     1-164#
CSEMIL    1-152     1-163#
CSEMIS    1-157     1-168#
CSEMIV    1-158     1-169#
CSEMNF    1-156     1-167#
CSEPRO    1-154     1-165#
CSIERR    1-152#   10-36
CSIMSG    1-25     10-34#
CTLRTN   52-33     52-42#
CTRLC     1-64      9-73     9-74     24-22    26-33    35-30    38-47    49-39
CTRLX     1-64     38-49
CTRLZ     1-64      1-89     3-24     9-69     9-133    38-45
```

Cross reference table (CREF V05.04)

```
CURPRM    1-63    39-96*  40-16   40-23*  40-27*  43-79*
CVTLC     1-22    33-77   34-5    47-44   50-45   76-10   80-12#
CXTRMN    1-40     9-60   42-14   42-38   43-16   43-142
DELCHR    1-22    24-64   29-47   33-67   68-37   75-16#
DELUAC   24-15    26-12   26-19#
DFRREL    7-29    46-24   47-9#
DISSLE    1-40    33-41   78-26
DOCTRL   50-101   52-10#
DOSCHD    1-70     2-33    2-54    4-17   11-26   55-46*
DOSWIT    1-36    50-94
ECHO     51-38    51-51   53-25   54-27   57-19   63-21   63-29   69-10   76-9#   76-34
ECHO1    53-27    53-39   54-29   59-20   60-64   60-96   60-98   60-100  64-37   64-41   64-75   64-84
         64-91    64-96   64-98   76-10#  76-37   76-39   76-41   76-58
ECHO2    51-66    70-88   76-14   76-17#
ECHOR    76-12    76-16   76-18   76-20#
ECOCTL   55-24    55-64   58-16   58-26   59-16   60-35   61-19   62-11   76-32#
EMTADR    1-94     6-23   45-31
EMTBLK    1-88     2-6     2-9     2-16    2-50    3-6     3-9     3-19    3-40    4-5     4-13    8-5
          8-15     8-18    8-34    8-37    8-55    8-67    8-68
EMTPS     1-90     3-17*
EMTXIT    1-90     4-26    5-11    6-42    7-37    8-10    8-58    8-70
ENQTL     1-60    32-7    67-50
ERHAZL   60-76    60-118#
ERV100   60-79    60-117#
ERV52    60-69    60-116#
ESC       1-68    37-8    38-40   39-41   60-116  60-117
ESCFLG    1-70    29-36   33-98   47-38   65-64
ESCHK    12-40    20-14#
ESCRLF   20-44#   37-24   37-25
ESCTL    20-34#   37-21
ESESC    20-60#   37-23
ESFF     20-52#   37-22
ESHT     20-66#   37-26
ESRS1    20-30    20-40   20-48   20-56   20-62   20-70   20-81#
ESRS2    20-29    20-35   20-39   20-47   20-53   20-61   20-67   20-74#
ESRTN    20-37    20-45   20-55   20-69   20-75#
ESUAC    20-22#   34-33
ESXIT    20-77    20-83#
FAKCMP    1-88     4-16
FETCHR   24-35    24-60   27-37   29-31   47-35   59-18   64-57   68-30   73-17#  75-22   75-48
FF        1-75     1-89    2-20   20-52   37-7
FORCEX    1-66    55-68
FRKPRI    1-39    18-22
GCCKAC   33-97    34-10   35-4#
GCCKCC   33-81    34-12   35-13   35-17   35-22   35-30#
GCCKCE    35-5    35-21#
GCCKDE    35-7    36-6#
GCCKDS   33-80    35-12#
GCCKES   33-22    33-99   33-101  34-5#
GCEAC    35-15    35-26#
GCECHO   35-26    36-8    38-8#   47-45
GCEND    33-90    35-31   35-35   35-37   35-41   36-7    36-12#
GCEXIT   33-54    36-13   36-17   36-19   36-21#
GCH1     33-14#   33-65   33-68
GETCHR    3-23     7-35    8-42    9-121   9-127   33-10#
GETUCH    1-88     2-41    8-22    8-26
```

Cross reference table (CREF V05.04)

```
GOTCHR  48-32   49-9#
GTCFCH   9-49    9-91   33-20   39-14#  39-83
GTCM1   12-64   21-30#
GTLTTY   1-90    9-32    9-41    9-89
GTSLCH   1-38   33-53
GTSPAC   1-28   24-5    24-10#
HAZEL    1-55   60-50   60-77
HION     8-7    28-7    28-14#
HIPUT    6-41    8-23   12-18   19-10#
HIRTN    8-8     8-10#   8-19    8-25    8-29    8-46    8-60
ICPCR   52-55   53-9#   54-36
ICPCTC  52-45   55-9#
ICPCTD  52-46   56-10#
ICPCTG  52-49   57-9#
ICPCTO  52-57   58-9#
ICPCTR  52-60   59-9#
ICPCTU  52-63   60-9#
ICPCTX  52-66   61-9#
ICPCTZ  52-68   62-9#
ICPESC  52-69   63-9#
ICPLF   52-52   54-9#
ICPRUB  50-105  64-9#
ILSW2    1-44   49-32
ILWAIT  33-64   46-7#
IN$ACT   1-42    9-61   43-143
INCHR   50-81   51-71   53-32   69-7#
INFIN   67-10#  71-28   72-25
INITFL   1-71   49-13
INITLN   1-92   49-45
INSCHR  24-47   27-44   70-53   74-15#  75-51
INTPRI   1-69    1-70    1-79   18-51   18-57   19-47   46-44   47-20   47-58   70-48   73-45   74-38
        75-43   75-60
JSWLOC   1-71   25-32   25-35*
KILCHR  60-105  64-39   64-46   68-13#
KPAR6    1-86   18-41   18-41*  18-43*  19-40   19-40*  19-42*  73-38   73-38*  73-44*  74-31   74-31*
        74-37*
LABTIM   1-45   49-51
LACTIV   1-61    7-13    8-45   24-48*  27-20   27-45*  33-62   46-33   47-18   55-69*  70-25   70-64*
        75-70*  75-81
LAFSIZ   1-69   24-49*  27-46*  27-53*  51-44   55-59*  71-17*
LBRKCH   1-80   50-33
LBRKCQ   1-82   81-52   81-54*
LCBIT    1-68    1-79   25-33   55-51   80-14
LCOL     1-72   14-41*  16-13*  16-15*  16-21*  16-28   16-40*  47-26   60-42   60-84*  70-18
LESCHR   1-53   20-28   20-34   34-40*
LESRTN   1-53   20-18   34-34*
LF       1-67    9-93    9-123  16-50   20-46   20-54   24-61   37-9    38-21   38-60   44-36   53-16
        53-26   53-33   76-40
LF$IN    1-95   35-42   36-12
LF$OUT   1-95   12-69
LF$WRT   1-95   45-13
LFWLIM   1-83   24-50*  27-47*  29-15*  29-55*  51-59   55-58*  71-18*
LHIPCT   1-93   33-60*  46-37*
LIFUN   12-60   21-8#
LINBUF   1-62   68-25   70-46   73-29   73-51   74-21   74-44
LINCNT   1-66    8-57   24-32   24-57   27-29   29-28   47-33   51-47   51-61   55-54*  68-18   70-58*
```

```
        75-67*  75-77
LINCUR   1-76   47-26*  60-44   64-54   70-18*
LINEND   1-62   68-27   70-44   73-33   73-49   74-26   74-42
LINNXT   1-65   55-52   55-53   68-23   70-41   70-47*  70-65   73-22   75-41   75-59*
LINPNT   1-61   24-31   27-35   29-27   33-66   47-29   55-52*  75-27   75-29*
LINSIZ   1-66   55-55
LINSPC   1-47   55-55*  70-22   70-57*  75-66*
LITIME   1-95   33-61*  46-38*  67-45
LJSW     1-41    1-90    6-15    7-5     9-32    9-41    9-89   25-34*  33-41   33-73   34-9    55-51*
        57-9    61-9    63-11   77-12   78-13   78-26   80-14
LNMAP    1-60   49-18   81-20   81-32
LNPRIM   1-67   50-14   55-62   60-70   81-19
LNSPAC   1-64   24-16   24-21*  26-21   26-32*  38-31   50-42
LOGBLK   1-94   45-28   45-34*
LOGBUF   1-94   45-25
LOGCH1  44-19   44-22   44-35   44-37   45-9#
LOGCHN   1-94   45-28   45-28
LOGCHR   1-23   12-71   35-50   36-20   44-9#
LOGCR    1-23   35-51   44-33#
LOGEND   1-94   45-20
LOGFLG   1-95   12-69   35-42   36-12   45-13
LOGPTR   1-95   45-18   45-32*  45-41*
LOTBUF   1-72   18-46   19-45
LOTEND   1-72   18-44   19-43
LOTNXT   1-72   18-40   18-47*  19-39   19-46*  55-57   58-15
LOTPNT   1-75   55-57*  58-15*
LOTSIZ   1-37   55-56   58-14
LOTSPC   1-72    6-9    18-20   18-24   18-39*  19-31   19-38*  31-29   55-56*  58-14*  76-17
LRBFIL   1-78   23-10*  60-48   60-91   64-92
LRDTIM   1-80    8-67*  67-33*
LRTCHR   1-80    8-68*   8-69*  33-69*
LSCCA    1-62   35-36   55-38
LSNDCH   1-53   12-45*  20-38   20-75*
LSPACT   1-64   24-19   26-23   38-33   50-46
LSTACT   1-37   55-53*  59-17   64-56   70-65*
LSTATE   1-76    2-36    2-57    4-20   11-29   58-18   67-22
LSTPRM   1-91   43-66
LSW      1-41    7-19*  12-22   13-21   19-15   19-17   24-13   26-10   27-13   28-5    28-14   29-13
        33-14   33-29*  33-34   33-36*  46-7    46-9*   49-19   50-77*  51-20*  53-45*  54-41*  55-20*
        55-65*  55-67*  57-30*  60-15*  61-30*  62-17*  63-30   63-33*  63-36*  64-15*  65-21*  65-48
        65-62*
LSW10    1-52   46-18*  49-53*  50-15   50-18*  75-84*
LSW11    1-37   17-17   43-52*  43-102* 52-16   60-71
LSW2     1-41   12-31   16-29   16-46   25-5*   25-12*  25-24*  25-31*  25-42*  25-49*  25-56*  30-5*
        30-12*  35-48   36-18   38-10   54-13   60-30   64-24   64-28   71-22   72-19   76-13   80-12
LSW3     1-41    7-21*   9-113* 12-24   12-38   12-50   13-34   18-32   20-74*  20-81*  25-57*  25-64*
        33-13*  33-95   33-102* 34-41*  35-12   35-14   35-16*  36-6    38-12   43-38*  46-29*  47-16
        47-24*  47-25*  47-54*  49-27   50-75   50-78*  50-83*  50-85   50-89*  55-34   55-61*  55-63*
        55-70*  58-9    58-17*  58-25*  60-24*  60-25   60-40   64-34   64-38*  70-15   70-17*  70-66
        70-68*  71-24*  72-21*  76-11   76-53   76-55*
LSW4     1-34    1-41    2-18    6-37    7-26    7-28*   7-34*   8-9*    8-20    8-59    8-61*   9-24*
        9-30    9-56    9-71    9-95    9-97*   9-104   9-135   9-148*  12-16   13-27   24-24*  26-35*
        28-16*  33-45   33-49   33-85   35-21   35-34   35-46   36-16   39-53*  39-54*  39-61*  39-68*
        39-75*  39-82*  41-12   41-14   43-12*  43-22   43-25*  43-35*  43-42   43-49   43-88*  43-91*
        43-123* 43-140  50-57   51-32   53-22   54-24   55-36   55-45*  56-10   66-12   70-86*  78-28
        78-32
```

Cross reference table (CREF V05.04)

Cross reference table (CREF V05.04)

```
QNSPNX    1-74     2-37     2-58     4-21     11-30
QUECHR    1-26    14-45    16-8     16-16    16-22    16-32    16-37    16-51    16-55    17-11#   29-51    31-31
R$CFST    1-34    42-15*   42-39*
R$INST    1-91     9-61    43-17*   43-143
RBEND    76-9     76-53#
REGCHR   50-107   51-9#    52-42    52-43    52-44    52-47    52-48    52-50    52-51    52-53    52-54    52-56
         52-58    52-59    52-61    52-62    52-64    52-65    52-67    52-70    52-71    52-72    52-73    56-22
         57-25    61-25
RSSPAC    1-29    26-5     26-10#
RUBOUT    1-66    14-39    50-103
S$$RUN    1-56    67-39
S$INWT    1-77    46-40    67-31
S$OTFN    1-60    32-6
S$OTWT    1-73    31-13    58-18
S$TTFN    1-81    67-18
S$TTSC    1-81    67-16
SCACHK   38-8     50-66    77-12#
SCACHR   50-79    54-18    55-22    59-11    60-19    62-9     64-19    79-16#
SESCTL   34-24    37-7#    37-12
SESRTN   34-30    37-22#
SETC      1-90     7-30     8-62
SETERR    1-37     6-32
SETRBF    1-28    23-5     23-10#
SFWAC     1-29    27-5     27-10#
SFWL      1-30    29-5     29-10#
SIGBRK   50-17    50-37    81-51#
SIGWAT    1-26    31-12    46-28    81-13#
SILFET    1-35    48-26
SLCHK    27-18    29-20    51-25    53-13    63-9     67-14    78-13#   79-16
SPACE     1-78    60-48    64-83
SPCTTY    1-76    33-41    33-73    34-9     55-51    57-9     61-9     63-11    77-12    78-13
STOP      1-60     7-20     9-44     9-63     9-137    12-26    19-19    33-16    33-30    33-37    35-52    43-55
         43-145   46-10
STPFLG    1-71    49-25
STRACT    1-24    50-52    51-53    53-15    53-40    54-30    55-25    57-20    61-20    62-12    63-35    65-71
         71-12#
STRCHR   50-59    63-22    63-32    65-65    65-69    69-7     70-9#    71-13    72-15
STRSNG   51-27    51-39    57-13    61-13    63-15    72-10#   79-22
SUCF2     1-35    43-53
TAB       1-66    20-66    37-11    64-48    64-59
TRNSFL    1-68    12-52    14-20
TRNSTR    1-38    18-52    19-51
TRYCHR   31-28#   81-34
TSTTY     1-6#     1-23
TTCPL     1-35    70-74
TTCSCH    1-85     2-30     2-51    11-14    11-25
TTEOF     3-25     3-33#
TTFIN     2-46     2-72     3-31     3-38     3-43     4-4#
TTINCP    1-23    48-15#
TTREAD    1-25     3-6#
TTYIN     1-26     7-5#
TTYOUT    1-26     6-5#
TTZERO    3-18     3-37#
UHIMEM    1-85     5-6
UOTSTR    5-10     9-114    11-9#
URO       1-90     4-9*     5-5      6-36     7-36*    8-41*    8-44*    8-57*
```

Cross reference table (CREF V05.04)

```
VALADB    1-88     2-8      2-14     3-8      3-13     5-9      8-17     8-36     8-40
VINTIO    1-93    33-60    46-37
VQUAN1    1-81    33-61    46-38
VT100     1-84    60-50
VT2007    1-84    60-50
VT2008    1-84    60-50
VT52      1-84    60-50    60-74
VTSLCH    1-68    12-62
VVLSCH    1-50    50-73
VVPWCH    1-36    52-14
WINCHR    1-52    17-19    19-25
WINPRT    1-35    52-25
WRITTT    1-25     2-6#
XHIIN     1-27     8-34#
XHIOUT    1-27     8-15#
XHISET    1-27     8-5#
XRDTIM    1-27     8-67#
XTCC      8-50     8-62#
XTERCK    1-27     8-55#
```

Cross reference table (CREF V05.04)

...CM1	40-37	43-121	43-127	45-28								
...CM2	40-37	40-37	40-37	40-37	43-121	43-127	43-127	43-127	43-127	45-28	45-28	45-28
	45-28											
...CM3	43-24											
...CM5	40-37	43-121	43-127	45-28								
...CM7	40-37	43-127	45-28									
.PURGE	1-17#	43-24										
.READW	1-17#	40-37	43-127									
.REOPE	1-17#	43-121										
.WRITW	1-17#	45-28										
DISABL	1-99#	18-16	19-30	31-28	46-32	47-15	70-40	73-37	74-30	75-40		
ENABL	1-103#	18-51	18-57	19-47	46-44	47-20	47-58	70-48	73-45	74-38	75-43	75-60
OCALL	1-107#	17-19	19-25	33-53	49-34	49-45	50-94	52-25	70-74			
TTMAP	1-137#	18-41	19-40	73-38	74-31							
TTMAPX	1-128#											
UNMAP	1-142#	18-43	19-42	73-44	74-37							
UNMAPX	1-134#											