TSTTY -- TSX Terminal I/O routi MACRO V05.05 Wednesday 18-Jan-89 10:26
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) 1980,1981,1982,1983,1984,1985,1986,1987,1988.
                                ;  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,VCTRLT,DOCTLT
                                        .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$OTFN,ENQTL,STOP
                        .GLOBL  CORUSR,CFPNT,LACTIV,LINPNT,CFPSAV
                        .GLOBL  LINEND,LINBUF,MXCPRM,PRMPNT,LSCCA
                        .GLOBL  CURPRM,CFEND,CFCHAN,CFBUF,$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,$ALTER
                ;
                ;  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     #$FORM0,LSW4(R1);DOES HE WANT FF ON WRITE OF BLK 0?
000424  001404                        BEQ     16$             ;BR IF NOT
000426  112700  000000G               MOVB    #FF,R0          ;OUTPUT FF TO GET TO TOP OF FORM
000432  004737  002716'               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  002716'               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  000000G                 CALL    QNSPNX          ;REQUEVE JOB AND CALL JOB SCHEDULER
     59 000556  004737  000000G                 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  002716'                 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  002716'                 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  000004G        TTREAD: MOV     EMTBLK+4,R3     ;GET BUFFER ADDRESS
000626  010300                        MOV     R3,R0           ;VALIDATE ADDRESS OF BUFFER
000630  004737  000000G                CALL    VALADB
000634  013704  000006G                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  000000G                CALL    VALADB          ;VALIDATE IT
000652  032777  000000G 000000G        BIT     #CS$EOF,@CHNADR ;DID LAST READ GET END OF FILE?
000660  001407                        BEQ     2$              ;BRANCH IF NOT
000662  042777  000000G 000000G        BIC     #CS$EOF,@CHNADR ;RESET END OF FILE FLAG
000670  052737  000000G 000000G        BIS     #CFLAG,EMTPS    ;SET C-BIT IN USER'S PS
000676  000425                        BR      TTZERO
000700  005737  000002G        2$:     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  002716'                CALL    PUTCHR
000716  004737  006012'        1$:     CALL    GETCHR          ;GET NEXT INPUT CHAR
000722  120027  000000G                CMPB    R0,#CTRLZ       ;CTRL-Z ==> END OF FILE
000726  001406                        BEQ     TTEOF           ;BRANCH IF END OF FILE
000730  105700                4$:     TSTB    R0              ;IGNORE NULLS
000732  001771                        BEQ     1$
000734  004737  000000G                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  000000G 000000G 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  000004G                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$:     CLR     R0              ;STORE NULL TO FILL OUR BLOCK
000776  004737  000000G                CALL    PUTUCH          ;MOVE NULL TO USER'S BUFFER
001002  077413                        SOB     R4,1$           ;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  000004G                 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  000000G         2$:     MOV     R4,URO          ;RETURN # WORDS IN USER'S R0
                                        ;
                                        ;  See if we need to call user's completion routine.
                                        ;
001024  023727  000010G 000001          CMP     EMTBLK+10,#1    ;COMPLETION ROUTINE SPECIFIED?
001032  101415                          BLOS    1$              ;BR IF NOT
                                        ;  Enter a request to call user's completion routine.
001034  004737  000000G                 CALL    FAKCMP          ;QUEUE A COMPLETION REQUEST
001040  105737  000000G                 TSTB    DOSCHD          ;IS JOB SCHEDULER CYCLE NEEDED?
001044  001410                          BEQ     1$              ;BR IF NOT
001046  004737  000000G                 CALL    CHKABT          ;SEE IF WE HAVE BEEN ABORTED
001052  016100  000000G                 MOV     LSTATE(R1),R0   ;GET JOB'S CURRENT EXECUTION STATE
001056  004737  000000G                 CALL    QNSPNX          ;REQUEUE JOB AND CALL JOB SCHEDULER
001062  004737  000000G                 CALL    CHKABT          ;SEE IF WE WERE ABORTED WHILE ASLEEP
                                        ;
                                        ;  Finished
                                        ;
001066  000137  000000G         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  002514'         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$:     MOV     @R0,R0          ;GET THE CHAR TO SEND
001202  032761  000000G 000000G         BIT     #$HITTY,LSW4(R1);ARE WE IN HIGH EFFICIENCY MODE?
001210  001003                          BNE     1$              ;BR IF YES
001212  004737  002716'                 CALL    PUTCHR          ;SEND THE CHAR
001216  000402                          BR      2$
001220  004737  003726'         1$:     CALL    HIPUT           ;SEND CHAR IN HIGH EFFICIENCY MODE
001224  000137  000000G         2$:     JMP     EMTXIT
```

```
                                        .SBTTL  .TTYIN
                                ;-------------------------------------------------------
                                ;  Process the .TTINR EMT
                                ;
001230  032761  000000G 000000G 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  000000G                 TST     LACTIV(R1)      ;ANY ACTIVATION CHARS PENDING?
001244  001042                         BNE     1$              ;BR IF YES
001246  004737  010030'                 CALL    CFTEST          ;IS INPUT COMING FROM A COMMAND FILE?
001252  103037                         BCC     1$              ;BRANCH IF YES
001254  032761  000000G 000000G        BIT     #$LOFCF,LSW9(R1);Are we processing a logoff command file?
001262  001405                         BEQ     4$              ;Br if not
001264  052761  000000G 000000G        BIS     #$DOOFF,LSW(R1) ;Force job logoff
001272  004737  000000G                 CALL    STOP            ;Stop job execution
001276  042761  000000G 000000G 4$:     BIC     #$NOIN,LSW3(R1) ;ALLOW TERMINAL INPUT TO OCCUR
001304  032761  000000G 000000G        BIT     #$DBGMD,LSW6(R1);IS DEBUGGER DOING TERMINAL INPUT?
001312  001004                         BNE     3$              ;BR IF YES -- WAIT FOR ACTIVATION CHAR
001314  032761  000000G 000000G        BIT     #$NOWTT,LSW5(R1);DID HE ENABLE NO-WAIT TT INPUT?
001322  001007                         BNE     2$              ;IF YES THEN SIGNAL NO CHARS AVAILABLE
001324  032761  000000G 000000G 3$:     BIT     #$FLAGC,LSW4(R1);HAVE WE ALREADY TOLD HIM ONCE?
001332  001007                         BNE     1$              ;BR IF YES (WAIT FOR INPUT NOW)
001334  052761  000000G 000000G        BIS     #$FLAGC,LSW4(R1);REMEMBER THAT WE HAVE TOLD HIM
001342  004737  011430'         2$:     CALL    DFRREL          ;RELEASE DEFERRED ECHO MODE
001346  000137  000000G                 JMP     SETC            ;RETURN WITH C-FLAG SET
                                ;
                                ;  Get a character
                                ;
001352  042761  000000G 000000G 1$:     BIC     #$FLAGC,LSW4(R1);HAVEN'T TOLD HIM NO CHARS AVAILABLE
001360  004737  006012'                 CALL    GETCHR          ;GO GET A CHARACTER
001364  010037  000000G                 MOV     R0,UR0          ;MOVE TO USER'S R0
001370  000137  000000G                 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  005430'              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$:     CALL    GETUCH          ;GET CHAR FROM USER'S BUFFER
001456  004737  003726'              CALL    HIPUT           ;SEND CHAR IN HIGH EFFICIENCY MODE
001462  077405                        SOB     R4,2$
001464  000754                        BR      HIRTN
001466  004737  000000G       1$:     CALL    GETUCH          ;GET CHAR FROM USER'S BUFFER
001472  004737  002716'              CALL    PUTCHR          ;SEND THE CHARACTER
001476  077405                        SOB     R4,1$
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  006012'       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  007332'          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  000000G 000000G          BIT     #$UKMON,LSW7(R1);IS USER COMMAND PROCESSOR ACTIVE?
     59 002010  001006                          BNE     11$             ;BR IF YES -- GET COMMAND FROM IT
     60 002012  013700  000000G                 MOV     CXTRMN,R0       ;GET ADDR OF SIMULATED RMON DATA FOR JOB
     61 002016  132760  000000G 000000G          BITB    #IN$ACT,R$INST(R0);IS INPUT BEING PROVIDED BY IND?
     62 002024  001457                          BEQ     3$              ;BR IF NOT
     63 002026  004737  000000G          11$:    CALL    STOP            ;REENTER KMON AND GET NEXT COMMAND FROM IND
     64                                     ;
     65                                     ;  We got a character from a control file.
     66                                     ;  See if character is a control-C or control-Z.
     67                                     ;  Note: we treat ctrl-Z the same as ctrl-C if KMON is reading file.
     68                                     ;
     69 002032  120027  000000G          8$:     CMPB    R0,#CTRLZ       ;IS CHAR CTRL-Z?
     70 002036  001006                          BNE     16$             ;BR IF NOT
     71 002040  032761  000000G 000000G          BIT     #$INKMN,LSW4(R1);IS KMON READING FILE?
     72 002046  001402                          BEQ     16$             ;BR IF NOT
     73 002050  112700  000000G                 MOVB    #CTRLC,R0       ;TRANSLATE CTRL-Z TO CTRL-C
     74 002054  120027  000000G          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  000000G 000000G          BIT     #GTLTTY,LJSW(R1);IS THIS A NON-TERMINATING .GTLIN?
     90 002070  001427                          BEQ     7$              ;BR IF NOT -- GO TERMINATE PROGRAM
     91 002072  004737  007332'          14$:    CALL    GTCFCH          ;SKIP TO END OF LINE THAT HAS ^C
     92 002076  103403                          BCS     15$
     93 002100  120027  000000G                 CMPB    R0,#LF          ;REACHED END OF LINE?
     94 002104  001372                          BNE     14$             ;LOOP IF NOT
     95 002106  032761  000000G 000000G 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  000000G 000000G          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  000000G 000000G 12$:    BIS     #$NTGCC,LSW9(R1);Push a control-C for next .GTLIN
    100 002134  013737  000000G 000000G          MOV     CFPNT,CFSPND    ;SAVE COMMAND FILE POINTER -- Suspend file
    101 002142  004737  010100'                 CALL    CFSTOP          ;Suspend command file input
    102 002146  000406                          BR      3$
    103 002150  110037  000000G          7$:     MOVB    R0,CFHOLD       ;PUSH COMMAND FILE CHAR
    104 002154  032761  000000G 000000G          BIT     #$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  000000G                 TST     CFPNT           ;Is input coming from a command file?
    112 002174  001003                          BNE     21$             ;Br if yes
    113 002176  042761  000000G 000000G          BIC     #$CTRLO,LSW3(R1);Reset control-O
    114 002204  004737  002514'          21$:    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     #$CFABT,LSW6(R1);If ctrl-C received, abort command files
002224  004737  006012'        19$:     CALL    GETCHR          ;GET AN INPUT CHAR
002230  042761  000000G 000000G         BIC     #$CFABT,LSW6(R1);Clear command file abort flag
002236  120027  000000G                 CMPB    R0,#LF          ;REACHED END OF LINE?
002242  001442                          BEQ     2$              ;BR IF YES
002244  120027  000041                  CMPB    R0,#'!          ;START OF COMMENT?
002250  001014                          BNE     17$             ;BR IF NOT
002252  005737  000000G                 TST     CFPNT           ;Input coming from command file?
002256  001004                          BNE     18$             ;If so, ! begins a comment
002260  032761  000000G 000000G         BIT     #$ALTER,LSW2(R1) ;Does user want !?
002266  001005                          BNE     17$             ;Br if so
002270  004737  006012'        18$:     CALL    GETCHR          ;SKIP OVER REST OF COMMENT
002274  120027  000000G                 CMPB    R0,#CR
002300  001373                          BNE     18$             ;LOOP TILL WE REACH END OF COMMENT
002302  020305                  17$:    CMP     R3,R5           ;ARE PAST THE END OF THE BUFFER?
002304  101341                          BHI     1$              ;BR IF YES (DISCARD THE CHARACTER)
002306  110023                          MOVB    R0,(R3)+        ;MOVE CHAR TO BUFFER
002310  120027  000003                  CMPB    R0,#3           ;is this a control-C
002314  001004                          BNE     4$              ;BR if not
002316  032761  000000G 000000G         BIT     #$ALTER,LSW2(R1);Should we activate on control-C
002324  001014                          BNE     6$              ;BR if yes
002326  120027  000000G         4$:     CMPB    R0,#CTRLZ       ;IS CHAR CTRL-Z?
002332  001326                          BNE     1$              ;BR IF NOT
002334  032761  000000G 000000G         BIT     #$INKMN,LSW4(R1);IS KMON DOING .GTLINE?
002342  001722                          BEQ     1$              ;BR IF NOT KMON
002344  004737  000000G                 CALL    STOP            ;TERMINATE PROGRAM IF CTRL-Z HIT
002350  020304                  2$:     CMP     R3,R4           ;IGNORE THE LF IF IT IS AT START OF LINE
002352  001716                          BEQ     1$
002354  105043                          CLRB    -(R3)           ;REPLACE CR WITH NULL
                                        ;
                                        ;  Finished getting input line
                                        ;
002356  013700  000000G         6$:     MOV     CFSPND,R0       ;GET @FILE POINTER
002362  001404                          BEQ     5$              ;BR IF NO NEED TO REPLACE
002364  004737  010124'                 CALL    CFSTRT          ;Restart command file input
002370  005037  000000G                 CLR     CFSPND          ;SAY THERE IS NO SUSPENDED COMMAND FILE
002374  042761  000000G 000000G 5$:     BIC     #$GTLIN,LSW4(R1);SAY .GTLIN EMT IS FINISHED
002402  012605                          MOV     (SP)+,R5
002404  012604                          MOV     (SP)+,R4
002406  012603                          MOV     (SP)+,R3
002410  012602                          MOV     (SP)+,R2
002412  012601                          MOV     (SP)+,R1
002414  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).
                                ;
002416  010046                  OUTSTR: MOV     R0,-(SP)
002420  010146                          MOV     R1,-(SP)
002422  010246                          MOV     R2,-(SP)
002424  113701  000000G                 MOVB    CORUSR,R1       ;GET USER'S INDEX #
002430  112200                  1$:     MOVB    (R2)+,R0        ;GET NEXT CHAR OF MESSAGE
002432  001406                          BEQ     2$              ;BRANCH WHEN END HIT
002434  120027  000200                  CMPB    R0,#200         ;STOP WITHOUT CR-LF?
002440  001413                          BEQ     9$              ;BRANCH IF YES
002442  004737  002716'                 CALL    PUTCHR          ;OUTPUT THE CHARACTER
002446  000770                          BR      1$
002450  112700  000015          2$:     MOVB    #15,R0          ;PUT OUT CR-LF
002454  004737  002716'                 CALL    PUTCHR
002460  112700  000012                  MOVB    #12,R0
002464  004737  002716'                 CALL    PUTCHR
002470  012602                  9$:     MOV     (SP)+,R2
002472  012601                          MOV     (SP)+,R1
002474  012600                          MOV     (SP)+,R0
002476  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.
                                ;
002500                          CSIMSG:
002500  006302                          ASL     R2              ;MULTIPLY BY TWO
002502  016202  000004'                  MOV     CSIERR(R2),R2   ;INDEX TO THE CORRECT ERROR STRING
002506  004737  002416'                  CALL    OUTSTR          ;PRINT THE ERROR MESSAGE
002512  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).
                                ;
002514  010046                  UOTSTR: MOV     R0,-(SP)
002516  010146                          MOV     R1,-(SP)
002520  010246                          MOV     R2,-(SP)
002522  010346                          MOV     R3,-(SP)
002524  010446                          MOV     R4,-(SP)
002526  012704  000000G               MOV     #TTCSCH,R4      ;SET CHARACTER COUNTER FOR SCHEDULING CALL
002532  113701  000000G               MOVB    CORUSR,R1       ;GET USER'S INDEX #
002536  012703  000200                MOV     #200,R3         ;GET CHAR USED TO STOP STRING WITHOUT CR-LF
002542  032702  000001                BIT     #1,R2           ;IS BUFFER ON EVEN OR ODD BYTE BOUNDARY?
002546  001420                      BEQ     3$              ;BR IF ON EVEN BYTE BOUNDARY
002550  005302                      DEC     R2              ;POINT TO 1ST BYTE OF WORD
002552  106522                      MFPD    (R2)+           ;GET WORD CONTAINING BYTE
002554  000426                      BR      4$
                                ;
                                ;  See if we need to interrupt output processing for a job scheduling cycle.
                                ;
002556  012704  000000G           1$:     MOV     #TTCSCH,R4      ;RESET THE CHARACTER COUNT FOR SCHEDULING CALL
002562  105737  000000G                   TSTB    DOSCHD          ;IS JOB SCHEDULER CYCLE NEEDED?
002566  001410                      BEQ     3$              ;BR IF NOT
002570  004737  000000G               CALL    CHKABT          ;SEE IF WE HAVE BEEN ABORTED
002574  016100  000000G               MOV     LSTATE(R1),R0   ;GET JOB'S CURRENT EXECUTION STATE
002600  004737  000000G               CALL    QNSPNX          ;REQUEUE JOB AND CALL JOB SCHEDULER
002604  004737  000000G               CALL    CHKABT          ;SEE IF WE WERE ABORTED WHILE ASLEEP
002610  005304                  3$:     DEC     R4              ;TIME TO CHECK FOR SCHEDULER CYCLE?
002612  003761                      BLE     1$              ;BR IF YES
002614  106522                      MFPD    (R2)+           ;GET WORD WITH NEXT 2 BYTES OF STRING
002616  111600                      MOVB    (SP),R0         ;GET NEXT BYTE OF STRING
002620  001417                      BEQ     5$              ;BR IF NULL AT END OF STRING HIT
002622  120003                      CMPB    R0,R3           ;IS CHAR #200?
002624  001413                      BEQ     10$             ;IF YES THEN END STRING WITHOUT CRLF
002626  004737  002716'               CALL    PUTCHR          ;SEND CHAR TO TERMINAL
002632  012600                  4$:     MOV     (SP)+,R0        ;GET WORD WITH CHAR IN HIGH-ORDER BYTE
002634  105000                      CLRB    R0              ;CLEAR LOW-ORDER BYTE
002636  000300                      SWAB    R0              ;MOVE HIGH-ORDER BYTE TO LOW-ORDER
002640  001410                      BEQ     2$              ;BR IF NULL AT END OF STRING HIT
002642  120003                      CMPB    R0,R3           ;IS THIS 200 AT END OF STRING?
002644  001416                      BEQ     9$              ;BR IF YES
002646  004737  002716'               CALL    PUTCHR          ;SEND CHAR TO TERMINAL
002652  000756                      BR      3$              ;GO SEND MORE CHARS
002654  005726                  10$:    TST     (SP)+           ;CLEAN OFF STACK
002656  000411                      BR      9$
002660  005726                  5$:     TST     (SP)+           ;CLEAN OFF STACK
002662  112700  000015          2$:     MOVB    #15,R0          ;PUT OUT CR-LF
002666  004737  002716'               CALL    PUTCHR
002672  112700  000012                MOVB    #12,R0
002676  004737  002716'               CALL    PUTCHR
002702  012604                  9$:     MOV     (SP)+,R4
002704  012603                          MOV     (SP)+,R3
002706  012602                          MOV     (SP)+,R2
```

```
     58 002710  012601                          MOV     (SP)+,R1
     59 002712  012600                          MOV     (SP)+,R0
     60 002714  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.
                                ;
002716                          PUTCHR:
                                ;
                                ;  If line is in "high efficiency" mode, bypass a lot of normal
                                ;  processing and use the high efficiency version of PUTCHR.
                                ;
002716  032761  000000G 000000G         BIT     #$HITTY,LSW4(R1);Is this line in high efficiency mode?
002724  001402                         BEQ     1$              ;Br if not
002726  000137  003726'                JMP     HIPUT           ;Use high efficiency version of PUTCHR
                                ;
                                ;  Return immediately if the line has disconnected
                                ;
002732  032761  000000C 000000G 1$:    BIT     #<$DISCN!$CTRLC>,LSW(R1) ;Has job disconnected?
002740  001406                         BEQ     2$              ;Br if not
002742  032761  000000G 000000G         BIT     #$NOIN,LSW3(R1) ;Are we doing logoff processing now?
002750  001057                         BNE     9$              ;Br if yes -- return immediately from PUTCHR
002752  004737  000000G               CALL    STOP            ;Terminate execution of job and enter KMON
                                ;
                                ;  Mask character to 7 or 8 bits depending on setting of EIGHTBIT option.
                                ;
002756  042700  177400          2$:    BIC     #^C<377>,R0     ;Mask character to 8 bits
002762  032761  000000G 000000G         BIT     #$8BIT,LSW2(R1) ;Is eightbit support option selected?
002770  001002                         BNE     8$              ;Br if yes
002772  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.
                                ;
002776  032761  000000G 000000G 8$:    BIT     #$NOOUT,LSW3(R1);Is output echo suppression in effect?
003004  001403                         BEQ     3$              ;Br if not
003006  004737  004106'               CALL    ESCHK           ;See if we should discard this character
003012  103436                         BCS     9$              ;Br if we should discard this character
                                ;
                                ;  Remember the last character output to this line
                                ;
003014  110061  000000G          3$:    MOVB    R0,LSNDCH(R1)   ;Remember last character sent to line
                                ;
                                ;  If character is being sent in transparent mode, set transparent
                                ;  flag for this character
                                ;
003020  032761  000000G 000000G         BIT     #$TRNSP,LSW3(R1);Is line is transparency mode?
003026  001403                         BEQ     4$              ;Br if not
003030  052700  000000G               BIS     #TRNSFL,R0      ;Set transparency flag for this character
003034  000415                         BR      5$
                                ;
                                ;  Determine if this character is a leadin character or part of
                                ;  a lead-in function sequence.
                                ;
```

```
     58 003036  005761  000000G          4$:     TST     LTSCMD(R1)        ;Are we processing a lead-in function now?
     59 003042  001403                           BEQ     6$                ;Br if not
     60 003044  004737  004304'                  CALL    LIFUN             ;Process the lead-in function
     61 003050  000417                           BR      9$                ;Finished with character
     62 003052  120037  000000G          6$:     CMPB    R0,VTSLCH         ;Is this the lead-in character?
     63 003056  001004                           BNE     5$                ;Br if not
     64 003060  012761  004340' 000000G         MOV     #GTCM1,LTSCMD(R1);Say we are starting a lead-in function
     65 003066  000410                           BR      9$                ;Finished with character
     66                                ;
     67                                ;  See if we should write this character to the terminal log file
     68                                ;
     69 003070  032737  000000G 000000G 5$:     BIT     #LF$OUT,LOGFLG    ;Are we logging output characters?
     70 003076  001402                           BEQ     7$                ;Br if not
     71 003100  004737  010776'                  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 003104  004737  003112'          7$:     CALL    PUTCH1            ;Queue character for terminal
     77                                ;
     78                                ;  Finished
     79                                ;
     80 003110  000207                   9$:     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.
                                ;
003112                          PUTCH1:
                                ;
                                ;  If this is a detached job, discard its output
                                ;
003112  032761  000000G 000000G         BIT     #$DETCH,LSW(R1) ;Is this a detached job?
003120  001015                          BNE     9$              ;Br if yes -- discard its output
                                ;
                                ;  If we are in a command file and program output is being suppressed,
                                ;  discard this character.
                                ;
003122  032761  000000G 000000G         BIT     #$CFSOT,LSW4(R1);Should we suppress command file output?
003130  001403                          BEQ     1$              ;Br if not
003132  005737  0000000               TST     CFPNT           ;Is input coming from a command file?
003136  001006                          BNE     9$              ;Br if yes -- discard this character
                                ;
                                ;  If Control-O has been typed, discard this character
                                ;
003140  032761  000000G 000000G 1$:    BIT     #$CTRLO,LSW3(R1);Has ctrl-O been typed?
003146  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
                                ;
003150  004737  003156'                 CALL    PUTCH2          ;Queue character for the terminal
                                ;
                                ;  Finished
                                ;
003154  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.
                                ;
003156  010246                 PUTCH2: MOV     R2,-(SP)
                                ;
                                ;  If character is being sent in transparency mode, by pass
                                ;  keeping track of its position.
                                ;
003160  032700  000000G                 BIT     #TRNSFL,R0      ;Is character in transparency mode?
003164  001015                          BNE     2$              ;Br if yes
                                ;
                                ;  Determine if this is a regular or control character.
                                ;
003166  020027  000037          3$:     CMP     R0,#37          ;Is this a control character?
003172  101005                          BHI     1$              ;Br if not
                                ;
                                ;  This is a control character.
                                ;  Call appropriate routine to process it.
                                ;
003174  010002                          MOV     R0,R2           ;Get character
003176  006302                          ASL     R2              ;Convert character to word table index
003200  004772  003230'                 CALL    @PC2VEC(R2)     ;Call processing routine
003204  000407                          BR      9$              ;Finished with character
                                ;
                                ;  This is a regular character.
                                ;  Keep track of cursor position.
                                ;
003206  120027  000000G         1$:     CMPB    R0,#RUBOUT      ;Is this a rubout character?
003212  001402                          BEQ     2$              ;Br if yes
003214  105261  000000G                 INCB    LCOL(R1)        ;Advance cursor column position
                                ;
                                ;  Insert this character into the terminal buffer
                                ;
003220  004737  003516'         2$:     CALL    QUECHR          ;Insert character into terminal buffer
                                ;
                                ;  Finished
                                ;
003224  012602                  9$:     MOV     (SP)+,R2
003226  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
                                ;
003230  003330'         PC2VEC: .WORD   PCCINS          ; 00 - NUL
003232  003330'                 .WORD   PCCINS          ; 01 - SOH
003234  003330'                 .WORD   PCCINS          ; 02 - STX
003236  003330'                 .WORD   PCCINS          ; 03 - ETX
003240  003330'                 .WORD   PCCINS          ; 04 - EOT
003242  003330'                 .WORD   PCCINS          ; 05 - ENQ
003244  003330'                 .WORD   PCCINS          ; 06 - ACK
003246  003330'                 .WORD   PCCINS          ; 07 - BEL
003250  003336'                 .WORD   PCCBS           ; 10 - BS
003252  003370'                 .WORD   PCCHT           ; 11 - HT
003254  003330'                 .WORD   PCCINS          ; 12 - LF
003256  003330'                 .WORD   PCCINS          ; 13 - VT
003260  003454'                 .WORD   PCCFF           ; 14 - FF
003262  003356'                 .WORD   PCCCR           ; 15 - CR
003264  003330'                 .WORD   PCCINS          ; 16 - SO
003266  003330'                 .WORD   PCCINS          ; 17 - SI
003270  003330'                 .WORD   PCCINS          ; 20 - DLE
003272  003330'                 .WORD   PCCINS          ; 21 - DC1
003274  003330'                 .WORD   PCCINS          ; 22 - DC2
003276  003330'                 .WORD   PCCINS          ; 23 - DC3
003300  003330'                 .WORD   PCCINS          ; 24 - DC4
003302  003330'                 .WORD   PCCINS          ; 25 - NAK
003304  003330'                 .WORD   PCCINS          ; 26 - SYN
003306  003330'                 .WORD   PCCINS          ; 27 - ETB
003310  003330'                 .WORD   PCCINS          ; 30 - CAN
003312  003330'                 .WORD   PCCINS          ; 31 - EM
003314  003330'                 .WORD   PCCINS          ; 32 - SUB
003316  003330'                 .WORD   PCCINS          ; 33 - ESC
003320  003330'                 .WORD   PCCINS          ; 34 - FS
003322  003330'                 .WORD   PCCINS          ; 35 - GS
003324  003330'                 .WORD   PCCINS          ; 36 - RS
003326  003330'                 .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.
                                ;
003330  004737  003516'       PCCINS: CALL    QUECHR          ;Put char into terminal buffer
003334  000207                        RETURN
                                ;
                                ;  Backspace -- Backup cursor position
                                ;
003336  105361  000000G       PCCBS:  DECB    LCOL(R1)        ;Backup cursor position
003342  002002                        BGE     1$              ;Br if didn't go past front of line
003344  105061  000000G               CLRB    LCOL(R1)        ;Don't allow to go to left of line start
003350  004737  003516'       1$:     CALL    QUECHR          ;Queue the character
003354  000207                        RETURN
                                ;
                                ;  Carriage return
                                ;
003356  105061  000000G       PCCCR:  CLRB    LCOL(R1)        ;Say cursor is at left margin
003362  004737  003516'               CALL    QUECHR          ;Queue the character
003366  000207                        RETURN
                                ;
                                ;  Tab -- Translate to spaces if tab simulation wanted
                                ;
003370  010246               PCCHT:  MOV     R2,-(SP)
003372  116102  000000G               MOVB    LCOL(R1),R2     ;Get current column position
003376  032761  000000G 000000G       BIT     #$TAB,LSW2(R1)  ;Should we simulate tabs?
003404  001011                        BNE     1$              ;Br if not
003406  112700  000040               MOVB    #' ,R0          ;Get space character
003412  004737  003516'       2$:     CALL    QUECHR          ;Send a space
003416  005202                        INC     R2              ;Advance cursor position
003420  032702  000007               BIT     #7,R2           ;Are we up to next tab stop?
003424  001372                        BNE     2$              ;Loop if not
003426  000406                        BR      9$
003430  004737  003516'       1$:     CALL    QUECHR          ;Send the tab character to the terminal
003434  062702  000010               ADD     #8.,R2          ;Bound up to next tab stop
003440  042702  000007               BIC     #7,R2
003444  110261  000000G       9$:     MOVB    R2,LCOL(R1)     ;Save new cursor position
003450  012602                        MOV     (SP)+,R2
003452  000207                        RETURN
                                ;
                                ;  Form feed -- See if we should translate to spaces
                                ;
003454  032761  000000G 000000G PCCFF:  BIT     #$FORM,LSW2(R1) ;Should we simulate form-feeds?
003462  001012                        BNE     2$              ;Br if not
003464  010246                        MOV     R2,-(SP)
003466  012702  000010               MOV     #8.,R2          ;Put out 8 LF characters
003472  112700  000000G               MOVB    #LF,R0          ;Get line feed character
003476  004737  003516'       1$:     CALL    QUECHR          ;Send a line feed
003502  077203                        SOB     R2,1$           ;Loop till all line feeds sent
003504  012602                        MOV     (SP)+,R2
003506  000402                        BR      9$
003510  004737  003516'       2$:     CALL    QUECHR          ;Send FF character to terminal
003514  000207                9$:     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.
                                ;
003516                          QUECHR:
                                ;
                                ;  See if display windowing is turned on for this line
                                ;
003516  005761  000000G                 TST     LWINDO(R1)      ;Is job doing display windowing?
003522  001410                          BEQ     1$              ;Br if not
003524  032761  000000G 000000G         BIT     #$NOWIN,LSW11(R1);Are we suppressing windowing now?
003532  001004                          BNE     1$              ;Br if yes
003534                                  OCALL   WINCHR          ;Pass character to window manager
003542  103402                          BCS     9$              ;Br if we should not display this char
                                ;
                                ;  Queue the character for transmission
                                ;
003544  004737  003552'         1$:     CALL    BUFCHR          ;Queue character for transmission
                                ;
                                ;  Finished
                                ;
003550  000207                  9$:     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.
                                ;
003552  010246                  BUFCHR: MOV     R2,-(SP)
                                ;
                                ;  Disable interrupts
                                ;
003554                          1$:     DISABL                  ;** Disable **
                                ;
                                ;  See if there is room in the output buffer for the character
                                ;
003562  026127  000000G 000010          CMP     LOTSPC(R1),#8.  ;Is there plenty of free space in buffer?
003570  101016                          BHI     2$              ;Br if yes
003572  105737  000000G                 TSTB    FRKPRI          ;Are we running at interrupt level? (echoing)
003576  001404                          BEQ     3$              ;Br if not at interrupt level
003600  005761  000000G                 TST     LOTSPC(R1)      ;Allow char echoing to use all of buffer
003604  003010                          BGT     2$              ;Br if buffer not completely full
003606  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.
                                ;
003610  004737  005730'          3$:     CALL    PCSPND          ;Suspend execution of job ** Enable **
003614  032761  000000G 000000G         BIT     #$CTRLO,LSW3(R1);Was Ctrl-O typed while we were asleep?
003622  001754                          BEQ     1$              ;Br if not
003624  000433                          BR      9$              ;Discard character if ctrl-O typed
                                ;
                                ;  There is room for the character in the output buffer
                                ;  Insert character into the buffer
                                ;
003626  005361  000000G          2$:     DEC     LOTSPC(R1)      ;Decrease free space count for buffer
003632  016102  000000G                 MOV     LOTNXT(R1),R2   ;Get pointer to next free cell in buffer
003636                                  TTMAP                   ;Map kernel PAR 6 to TT buffer
003652  110022                          MOVB    R0,(R2)+        ;Move character into TT buffer
003654                                  UNMAP                   ;Remap kernel PAR 6 to job context block
003662  020261  000000G                 CMP     R2,LOTEND(R1)   ;Did we move beyond end of buffer?
003666  103402                          BLO     4$              ;Br if not
003670  016102  000000G                 MOV     LOTBUF(R1),R2   ;Wrap around to front of buffer
003674  010261  000000G          4$:     MOV     R2,LOTNXT(R1)   ;Save next-character pointer
                                ;
                                ;  Start transmitter for this line
                                ;
003700                                  ENABL                   ;** Enable **
003706  004777  000000G                 CALL    @TRNSTR         ;Start transmitter for this line
003712  000403                          BR      10$
                                ;
                                ;  Finished
                                ;
003714                          9$:     ENABL                   ;** Enable **
```

```
     58 003722  012602                    10$:    MOV     (SP)+,R2
     59 003724  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.
                                ;
003726                          HIPUT:;
003726  010246                          MOV     R2,-(SP)
                                ;
                                ;  If this is a detached job, discard its output
                                ;
003730  032761  000000G 000000G         BIT     #$DETCH,LSW(R1) ;Is this a detached job?
003736  001061                          BNE     9$              ;Br if yes -- discard its output
003740  032761  000000C 000000G         BIT     #<$DISCN+$CTRLC+$DETCH>,LSW(R1) ;IS LINE DISCONNECTED?
003746  001402                          BEQ     1$              ;BR IF NOT
003750  004737  000000G                 CALL    STOP            ;STOP JOB
                                ;
                                ;  See if display windowing is turned on for this line
                                ;
003754  005761  000000G         1$:     TST     LWINDO(R1)      ;Is job doing display windowing?
003760  001404                          BEQ     4$              ;Br if not
003762                                  OCALL   WINCHR          ;Pass character to window manager
003770  103444                          BCS     9$              ;Br if we should not display this char
                                ;
                                ;  See if there is room in the output buffer for this character
                                ;
003772                          4$:     DISABL                  ;** DISABLE **
004000  026127  000000G 000010         CMP     LOTSPC(R1),#8.  ;ENOUGH SPACE IN OUTPUT BUFFER?
004006  101003                          BHI     2$              ;BR IF PLENTY OF SPACE
004010  004737  005730'                 CALL    PCSPND          ;SUSPEND PROGRAM TILL SPACE IS AVAILABLE
004014  000766                          BR      4$              ;NOW GO TRY AGAIN
                                ;
                                ;  Insert character into output buffer
                                ;
004016  005361  000000G         2$:     DEC     LOTSPC(R1)      ;SAY LESS SPACE AVAILABLE IN BUFFER
004022  016102  000000G                 MOV     LOTNXT(R1),R2   ;GET POINTER TO FREE CELL IN BUFFER
004026                                  TTMAP                   ;MAP TO TT BUFFER AREA
004042  110022                          MOVB    R0,(R2)+        ;MOVE CHAR INTO TT BUFFER
004044                                  UNMAP
004052  020261  000000G                 CMP     R2,LOTEND(R1)   ;HAVE WE HIT THE END OF THE TT BUFFER?
004056  103402                          BLO     3$              ;BR IF NOT
004060  016102  000000G                 MOV     LOTBUF(R1),R2   ;WRAPAROUND TO FRONT OF BUFFER
004064  010261  000000G         3$:     MOV     R2,LOTNXT(R1)   ;SAVE NEW BUFFER POINTER
004070                                  ENABL                   ;** ENABLE **
                                ;
                                ;  Try to start transmission to line
                                ;
004076  004777  000000G                 CALL    @TRNSTR         ;TRY TO START TRANSMISSION TO LINE
                                ;
                                ;  Finished
                                ;
004102  012602                  9$:     MOV     (SP)+,R2
004104  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.
                                ;
004106                          ESCHK:
                                ;
                                ;  Jump to appropriate restart routine based on echo suppression class
                                ;
004106  000171  000000G                 JMP     @LESRTN(R1)     ;Jump to echo suppression restart routine
                                ;
                                ;  Restart after a normal character
                                ;
004112  010046                  ESUAC:  MOV     R0,-(SP)        ;PUSH CHARACTER BEING SENT
004114  020027  000141                  CMP     R0,#141         ;IS IT A LOWER CASE LETTER?
004120  103405                          BLO     1$              ;BR IF NOT
004122  020027  000172                  CMP     R0,#172
004126  101002                          BHI     1$
004130  162716  000040                  SUB     #40,(SP)        ;CONVERT LOWER-CASE TO UPPER-CASE
004134  122661  000000G         1$:     CMPB    (SP)+,LESCHR(R1);IS CHAR BEING SENT SAME AS ECHO-SUPPRESSION CHAR?
004140  001445                          BEQ     ESRS2           ;IF YES THEN RESTART WITH NEXT CHAR
004142  000453                          BR      ESRS1           ;IF NOT RESTART IMMEDIATELY
                                ;
                                ;  Restart output after control type activation chars (ctrl-x etc.)
                                ;
004144  120061  000000G         ESCTL:  CMPB    R0,LESCHR(R1)   ;CONTROL CHAR ECHOED BACK?
004150  001441                          BEQ     ESRS2           ;IF YES RESTART WITH NEXT CHAR
004152  120027  000136                  CMPB    R0,#'^          ;ECHO ^-CHAR?
004156  001441                          BEQ     ESRTN           ;IF YES WAIT FOR NEXT CHAR
004160  126127  000000G 000136         CMPB    LSNDCH(R1),#'^  ;WAS PREVIOUS CHAR ^?
004166  001432                          BEQ     ESRS2           ;IF YES, RESTART WITH NEXT CHAR
004170  000440                          BR      ESRS1           ;RESTART WITH THIS CHAR
                                ;
                                ;  Restart output after cr or lf.
                                ;
004172  120027  000000G         ESCRLF: CMPB    R0,#CR          ;IS THIS CHAR CR?
004176  001431                          BEQ     ESRTN           ;YES -- KEEP SKIPPING
004200  120027  000000G                 CMPB    R0,#LF          ;IS IT LF?
004204  001423                          BEQ     ESRS2           ;YES -- RESTART WITH NEXT CHAR.
004206  000431                          BR      ESRS1           ;NOT CR OF LF.  RESTART IMMEDIATELY
                                ;
                                ;  Restart after form feed
                                ;
004210  120027  000000G         ESFF:   CMPB    R0,#FF          ;ECHO FORM FEED CHAR?
004214  001417                          BEQ     ESRS2           ;IF YES, RESTART WITH NEXT CHAR
004216  120027  000000G                 CMPB    R0,#LF          ;IGNORE A STRING OF LF'S
004222  001417                          BEQ     ESRTN
004224  000422                          BR      ESRS1
                                ;
```

```
                                        ;  Restart after escape.
                                        ;
004226  120027  000044          ESESC:  CMPB    R0,#'$          ;OK TO ECHO $ TOO.
004232  001410                          BEQ     ESRS2
004234  000416                          BR      ESRS1           ;RESTART IMMEDIATELY.
                                        ;
                                        ;  Restart after tab.
                                        ;
004236  120027  000000G         ESHT:   CMPB    R0,#TAB         ;ECHO TAB CHAR?
004242  001404                          BEQ     ESRS2           ;IF YES, RESTART WITH NEXT CHAR
004244  120027  000040                  CMPB    R0,#' '         ;IGNORE A STRING OF BLANKS
004250  001404                          BEQ     ESRTN
004252  000407                          BR      ESRS1
                                        ;
                                        ;  Set output restart with next character.
                                        ;
004254  042761  000000G 000000G ESRS2:  BIC     #$NOOUT,LSW3(R1);RESTART OUTPUT WITH NEXT CHAR
004262  110061  000000G         ESRTN:  MOVB    R0,LSNDCH(R1)   ;REMEMBER LAST CHAR SENT
004266  000261                          SEC                     ;SAY TO DISCARD THIS CHARACTER
004270  000404                          BR      ESXIT
                                        ;
                                        ;  Restart output immediately with this character.
                                        ;
004272  042761  000000G 000000G ESRS1:  BIC     #$NOOUT,LSW3(R1);SAY ECHO SUPPRESSION IS TURNED OFF
004300  000241                          CLC                     ;Say to restart immediately
004302  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
004304  010246                  LIFUN:  MOV     R2,-(SP)
004306  010346                          MOV     R3,-(SP)
004310  010446                          MOV     R4,-(SP)
004312  010546                          MOV     R5,-(SP)
                                ;
                                ;  Enter correct processing routine
                                ;
004314  016102  000000G                 MOV     LTSCMD(R1),R2   ;Get address of correct processing routine
004320  005061  000000G                 CLR     LTSCMD(R1)      ;Say no processing routine pending now
004324  004712                          CALL    (R2)            ;Enter processing routine
                                ;
                                ;  Finished
                                ;
004326  012605                          MOV     (SP)+,R5
004330  012604                          MOV     (SP)+,R4
004332  012603                          MOV     (SP)+,R3
004334  012602                          MOV     (SP)+,R2
004336  000207                          RETURN
                                ;
                                ;-----------------------------------------------------------------------
                                ;  Get letter after leadin character and identify command.
                                ;
004340  162700  000101          GTCM1:  SUB     #'A,R0          ;CONVERT LETTER TO TABLE INDEX
004344  100406                          BMI     9$              ;BR IF NOT LEGAL COMMAND
004346  020027  000032                  CMP     R0,#MAXCC       ;SEE IF IN LEGAL RANGE
004352  103003                          BHIS    9$              ;BR IF TOO BIG
004354  006300                          ASL     R0              ;MAKE INTO WORD TABLE INDEX
004356  004770  004364'                 CALL    @CCVECT(R0)     ;ENTER PROCESSING ROUTINE
                                ;
                                ;  Finished
                                ;
004362  000207                  9$:     RETURN
```

```
                                ;-----------------------------------------------------------------------------
                                ;  Define TSX command control characters
                                ;
004364  004450'         CCVECT: .WORD   CMDA             ;SET RUBOUT FILLER CHARACTER
004366  004466'                 .WORD   CMDB             ;ENABLE VT50 ESC-LETTER ACTIVATION
004370  004476'                 .WORD   CMDC             ;DISABLE VT50 ESC-LETTER ACTIVATION
004372  004506'                 .WORD   CMDD             ;DEFINE ACTIVATION CHARACTER
004374  004736'                 .WORD   CMDE             ;TURN ON ECHO
004376  004746'                 .WORD   CMDF             ;TURN OFF ECHO
004400  004756'                 .WORD   CMDG             ;NOP
004402  004760'                 .WORD   CMDH             ;DISABLE VIRTUAL LINES
004404  004770'                 .WORD   CMDI             ;ENABLE LOWER CASE INPUT
004406  005020'                 .WORD   CMDJ             ;DISABLE LOWER CASE INPUT
004410  005030'                 .WORD   CMDK             ;SET DEFERRED CHAR ECHO MODE
004412  005040'                 .WORD   CMDL             ;SET IMMEDIATE CHAR ECHO MODE
004414  005056'                 .WORD   CMDM             ;SET TRANSPARENCY MODE ON
004416  005066'                 .WORD   CMDN             ;SUSPEND COMMAND FILE INPUT
004420  005106'                 .WORD   CMDO             ;RESTART COMMAND FILE INPUT
004422  005126'                 .WORD   CMDP             ;RESET USER ACTIVATION CHAR
004424  005236'                 .WORD   CMDQ             ;SET ACTIVATION ON FIELD WIDTH
004426  005412'                 .WORD   CMDR             ;TURN ON HIGH-EFFICIENCY TTY MODE
004430  005450'                 .WORD   CMDS             ;TURN ON SINGLE CHARACTER ACTIVATION MODE
004432  005460'                 .WORD   CMDT             ;TURN OFF SINGLE CHARACTER ACTIVATION MODE
004434  005470'                 .WORD   CMDU             ;ENABLE NON-WAIT TT INPUT
004436  005500'                 .WORD   CMDV             ;SET FIELD WIDTH LIMIT
004440  005670'                 .WORD   CMDW             ;TURN TAPE MODE ON
004442  005700'                 .WORD   CMDX             ;TURN TAPE MODE OFF
004444  005710'                 .WORD   CMDY             ;DISABLE AUTO LF ECHO
004446  005720'                 .WORD   CMDZ             ;ENABLE AUTO LF ECHO
000032                  MAXCC   =       <.-CCVECT>/2     ;# COMMANDS
```

```
                                ;-------------------------------------------------------------------------------
                                ;  Command - A
                                ;  The next char will be the rubout filler character.
                                ;
004450  012761  004460' 000000G CMDA:   MOV     #SETRBF,LTSCMD(R1);SET NEXT PROCESSING ROUTINE
004456  000207                         RETURN
                                ;
                                ;  THIS CHAR IS THE RUBOUT FILLER CHARACTER.
                                ;
004460  110061  000000G          SETRBF: MOVB    R0,LRBFIL(R1)   ;SET NEW RUBOUT FILLER CHARACTER
004464  000207                         RETURN

                                ;-------------------------------------------------------------------------------
                                ;  Command - B
                                ;  Enable activation on vt50 escape-letter sequence.
                                ;
004466  052761  000000G 000000G CMDB:   BIS     #$VTESC,LSW5(R1);ENABLE THAT ACTIVATION SEQUENCE
004474  000207                         RETURN

                                ;-------------------------------------------------------------------------------
                                ;  Command - C
                                ;  Disable activation on vt50 escape-letter sequence.
                                ;
004476  042761  000000G 000000G CMDC:   BIC     #$VTESC,LSW5(R1);DISABLE ACTIVATION CLASS
004504  000207                         RETURN
```

```
                                ;-----------------------------------------------------------------------
                                ;  Command - D
                                ;  Define additional activation character (which follows).
                                ;
004506  012761  004516' 000000G CMDD:   MOV     #GTSPAC,LTSCMD(R1);SET PROCESSING ROUTINE
004514  000207                         RETURN
                                ;
                                ;  This character is a new activation character.
                                ;
004516  010346                  GTSPAC: MOV     R3,-(SP)
004520  010446                          MOV     R4,-(SP)
004522  010546                          MOV     R5,-(SP)
004524  032761  000000G 000000G        BIT     #$DETCH,LSW(R1) ;Is this a detached job?
004532  001075                         BNE     9$              ;Br if yes -- Ignore function
004534  004737  005154'               CALL    DELUAC          ;DEL CHAR IF ALREADY IN TABLE
004540  016102  000000G               MOV     LNSPAC(R1),R2   ;GET # OF CHARS DEFINED SO FAR
004544  020227  000000G               CMP     R2,#MXSPAC      ;DEFINED ALL ALLOWED?
004550  103066                         BHIS    9$              ;CAN'T HAVE ANY MORE
004552  066102  000000G               ADD     LSPACT(R1),R2   ;POINT TO FREE SPOT IN TABLE
004556  110012                         MOVB    R0,(R2)         ;STORE AWAY CHARACTER
004560  005261  000000G               INC     LNSPAC(R1)      ;SAY 1 MORE CHAR IN TABLE
004564  120027  000000G               CMPB    R0,#CTRLC       ;IS CTRL-C A SPECIAL ACTIV CHAR?
004570  001003                         BNE     1$              ;BRANCH IF NOT CTRL-C
004572  052761  000000G 000000G        BIS     #$UCTLC,LSW4(R1);REMEMBER SPECIAL CTRL-C
                                ;
                                ;  If user typed ahead any of the characters that are being declared
                                ;  to be an activation char, mark them as activation chars.
                                ;
004600  005003                  1$:     CLR     R3              ;Count chars in R3
004602  010005                          MOV     R0,R5           ;Carry new activation char in R5
004604  016102  000000G               MOV     LINPNT(R1),R2   ;Get pointer to next char in TT buffer
004610  020361  000000G         2$:     CMP     R3,LINCNT(R1)   ;Any more chars to check?
004614  103044                         BHIS    9$              ;Br if not
004616  010204                         MOV     R2,R4           ;Save current character pointer
004620  004737  016420'               CALL    FETCHR          ;Fetch a char from TT input buffer
004624  005203                         INC     R3              ;Count characters that we check
004626  120005                         CMPB    R0,R5           ;Is this the activation char?
004630  001367                         BNE     2$              ;Loop if not
                                ;
                                ;  We found the activation char in the type-ahead characters.
                                ;  Mark it as an activation character.
                                ;
004632  032700  000000G         4$:     BIT     #ACFLAG,R0      ;Is char already marked for activation?
004636  001013                         BNE     3$              ;Br if yes
004640  052700  000000G               BIS     #ACFLAG,R0      ;Set activation-character flag
004644  010402                         MOV     R4,R2           ;Get back pointer to character
004646  004737  016562'               CALL    INSCHR          ;Insert character in buffer
004652  005261  000000G               INC     LACTIV(R1)      ;Say another pending activation char
004656  005061  000000G               CLR     LAFSIZ(R1)      ;Clear field width activation
004662  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.
                                ;
004666  120027  000000G         3$:     CMPB    R0,#CR          ;Is activation character a carriage-return?
004672  001346                         BNE     2$              ;Br if not
004674  020361  000000G               CMP     R3,LINCNT(R1)   ;Any chars left in buffer?
```

```
     58 004700  103012                          BHIS    9$              ;Br if not
     59 004702  010204                          MOV     R2,R4           ;Save position of possible line feed
     60 004704  004737  016420'                 CALL    FETCHR          ;Fetch character that follows CR
     61 004710  120027  000000G                 CMPB    R0,#LF          ;Is that character a line feed?
     62 004714  001335                          BNE     2$              ;Br if not
     63 004716  010402                          MOV     R4,R2           ;Get back pointer to line feed
     64 004720  004737  016712'                 CALL    DELCHR          ;Delete the line feed
     65 004724  000731                          BR      2$
     66                                ;
     67                                ;  Finished
     68                                ;
     69 004726  012605                 9$:     MOV     (SP)+,R5
     70 004730  012604                         MOV     (SP)+,R4
     71 004732  012603                         MOV     (SP)+,R3
     72 004734  000207                         RETURN
```

```
                                ;-----------------------------------------------------------------------------
                                ;  Command - E
                                ;  Turn on character echoing.
                                ;
004736  052761  000000G 000000G CMDE:   BIS     #$ECHO,LSW2(R1) ;ENABLE ECHOING
004744  000207                        RETURN

                                ;-----------------------------------------------------------------------------
                                ;  Command - F
                                ;  Turn off character echoing.
                                ;
004746  042761  000000G 000000G CMDF:   BIC     #$ECHO,LSW2(R1) ;DISABLE ECHOING
004754  000207                        RETURN

                                ;-----------------------------------------------------------------------------
                                ;  Command - G
                                ;
004756  000207                  CMDG:   RETURN

                                ;-----------------------------------------------------------------------------
                                ;  Command - H
                                ;  Disable virtual lines.
                                ;
004760  052761  000000G 000000G CMDH:   BIS     #$NOVLN,LSW2(R1);DISABLE VIRTUAL LINES
004766  000207                        RETURN

                                ;-----------------------------------------------------------------------------
                                ;  Command - I
                                ;  Enable lower case input.
                                ;
004770  052761  000000G 000000G CMDI:   BIS     #$LC,LSW2(R1)   ;DO "SET TTY LC"
004776  106537  000000G               MFPD    @#JSWLOC        ;GET USER'S JSW VALUE
005002  052716  000000G               BIS     #LCBIT,(SP)     ;SET LOWER-CASE ENABLE FLAG
005006  011661  000000G               MOV     (SP),LJSW(R1)   ;SAVE JSW IN INTERNAL TABLE
005012  106637  000000G               MTPD    @#JSWLOC        ;STORE BACK INTO USER'S AREA
005016  000207                        RETURN

                                ;-----------------------------------------------------------------------------
                                ;  Command - J
                                ;  Disable lower case input.
                                ;
005020  042761  000000G 000000G CMDJ:   BIC     #$LC,LSW2(R1)   ;DO "SET TTY NOLC"
005026  000207                        RETURN

                                ;-----------------------------------------------------------------------------
                                ;  Command - K
                                ;  Set deferred character echo mode.
                                ;
005030  052761  000000G 000000G CMDK:   BIS     #$DEFER,LSW2(R1);SET DEFERRED ECHO FLAG
005036  000207                        RETURN

                                ;-----------------------------------------------------------------------------
                                ;  Command - L
                                ;  Set immediate mode char echo.
                                ;
005040  042761  000000G 000000G CMDL:   BIC     #$DEFER,LSW2(R1) ;Echo immediately from now on
005046  042761  000000C 000000G        BIC     #<$DODFR!$GCECO>,LSW3(R1) ;Say we are not now deferring
```

```
     58 005054  000207                          RETURN
     59
     60                                ;-----------------------------------------------------------------------
     61                                ;  Command - M
     62                                ;  Set output transparency mode
     63                                ;
     64 005056  052761  000000G 000000G CMDM:    BIS     #$TRNSP,LSW3(R1);TURN ON TRANSPARENCY MODE
     65 005064  000207                          RETURN
     66
     67                                ;-----------------------------------------------------------------------
     68                                ;  Command - N
     69                                ;  Suspend input from a control file
     70                                ;
     71 005066  013702  000000G          CMDN:   MOV     CFPNT,R2        ;IS A COMMAND FILE ACTIVE NOW?
     72 005072  001404                          BEQ     1$              ;BR IF NOT
     73 005074  010237  000000G                 MOV     R2,CFSPND       ;SAVE POINTER
     74 005100  004737  010100'                 CALL    CFSTOP          ;Suspend command file input
     75 005104  000207                  1$:     RETURN
     76
     77                                ;-----------------------------------------------------------------------
     78                                ;  Command - O
     79                                ;  Restart input from suspended command file
     80                                ;
     81 005106  013700  000000G          CMDO:   MOV     CFSPND,R0       ;IS COMMAND FILE SUSPENDED?
     82 005112  001404                          BEQ     1$              ;BR IF NOT
     83 005114  004737  010124'                 CALL    CFSTRT          ;Restart command file input
     84 005120  005037  000000G                 CLR     CFSPND
     85 005124  000207                  1$:     RETURN
```

```
                                ;-------------------------------------------------------------------------
                                ;  Command - P
                                ;  Reset user specified activation char.
                                ;
005126  012761  005136' 000000G CMDP:   MOV     #RSSPAC,LTSCMD(R1);SET PROCESSING ROUTINE
005134  000207                          RETURN
                                ;
                                ;  THIS CHAR IS THE CHAR TO RESET.
                                ;
005136  032761  000000G 000000G RSSPAC: BIT     #$DETCH,LSW(R1) ;Is this a detached job?
005144  001002                          BNE     9$              ;Br if yes -- Ignore function
005146  004737  005154'                CALL    DELUAC          ;CALL SUBROUTINE TO DEL CHAR FROM TABLE
005152  000207                  9$:     RETURN
                                ;
                                ;  SUBROUTINE TO DELETE A USER-DEFINED ACTIVATION FROM THE TABLE
                                ;  OF ACTIVATION CHARACTERS.
                                ;  WHEN CALLED THE CHARACTER TO BE DELETED MUST BE IN R0.
                                ;
005154  010246                  DELUAC: MOV     R2,-(SP)
005156  010346                          MOV     R3,-(SP)
005160  016102  000000G                 MOV     LNSPAC(R1),R2   ;GET # OF USER ACTIVATION CHARS
005164  001421                          BEQ     3$              ;BR IF NONE TO RESET
005166  016103  000000G                 MOV     LSPACT(R1),R3   ;POINT TO START OF TABLE
005172  120023                  2$:     CMPB    R0,(R3)+        ;SEARCH FOR CHAR
005174  001402                          BEQ     1$              ;BR WHEN FOUND
005176  077203                          SOB     R2,2$           ;LOOP IF MORE TO CHECK
                                ;  COULDN'T FIND CHAR
005200  000413                          BR      3$
                                ;  FOUND CHAR
005202  112363  177776          1$:     MOVB    (R3)+,-2(R3)    ;MOVE BACK REST OF CHARS
005206  077203                          SOB     R2,1$
005210  005361  000000G                 DEC     LNSPAC(R1)      ;SAY ONE LESS CHAR TO ACTIVATE ON
005214  120027  000000G                 CMPB    R0,#CTRLC       ;DE-ACTIVATING CTRL-C?
005220  001003                          BNE     3$              ;BR IF NOT
005222  042761  000000G 000000G        BIC     #$UCTLC,LSW4(R1);REMEMBER NO MORE CTRL-C
005230  012603                  3$:     MOV     (SP)+,R3
005232  012602                          MOV     (SP)+,R2
005234  000207                          RETURN
```

```
                                ;-----------------------------------------------------------------------
                                ;  Command - Q
                                ;  Set field width as an activation condition
                                ;
005236  012761  005246' 000000G CMDQ:   MOV     #SFWAC,LTSCMD(R1);SET PROCESSING ROUTINE
005244  000207                         RETURN
                                ;
                                ;  Interpret this char as value of field width
                                ;
005246  010246                 SFWAC:  MOV     R2,-(SP)
005250  010346                         MOV     R3,-(SP)
005252  010446                         MOV     R4,-(SP)
005254  032761  000000G 000000G        BIT     #$DETCH,LSW(R1) ;Is this a detached job?
005262  001047                         BNE     9$              ;Br if yes -- Ignore function
005264  010003                         MOV     R0,R3           ;CARRY FIELD WIDTH IN R3
005266  004737  010030'               CALL    CFTEST          ;INPUT COMING FROM COMMAND FILE?
005272  103043                         BCC     9$              ;BR IF YES
005274  004737  017362'               CALL    SLCHK           ;IS SINGLE LINE EDITOR IN CONTROL?
005300  103436                         BCS     1$              ;BR IF YES -- SET VALUE AND LET SL HANDLE IT
005302  005761  000000G               TST     LACTIV(R1)      ;IS ACTIVATION CHAR PENDING NOW?
005306  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
                                ;
005310  005703                         TST     R3              ;IS HE RESETTING FIELD WIDTH ACTIVATION?
005312  001431                         BEQ     1$              ;BR IF YES
005314  020361  000000G               CMP     R3,LINCNT(R1)   ;REACHED ACTIVATION ALREADY?
005320  101026                         BHI     1$              ;BR IF NOT
                                ;
                                ;  User has typed ahead enough to fill this field.
                                ;  Set activation flag in last char in field.
                                ;
005322  016102  000000G               MOV     LINPNT(R1),R2   ;POINT TO 1ST CHAR PENDING
005326  010204                 3$:     MOV     R2,R4           ;SAVE POINTER TO NEXT CHAR
005330  004737  016420'               CALL    FETCHR          ;FETCH THE NEXT INPUT CHARACTER
005334  103422                         BCS     9$              ;STRANGE -- NO MORE CHARS TO GET
005336  032700  000000G               BIT     #ACFLAG,R0      ;IS THIS CHAR ALREADY FLAGGED FOR ACTIVATION?
005342  001017                         BNE     9$              ;BR IF YES
005344  077310                         SOB     R3,3$           ;LOOP TILL WE GET LAST CHAR IN FIELD
005346  052700  000000G               BIS     #ACFLAG,R0      ;SET ACTIVATION-CHARACTER FLAG FOR CHAR
005352  010402                         MOV     R4,R2           ;GET BACK POINTER TO CHAR POS IN BUFFER
005354  004737  016562'               CALL    INSCHR          ;REINSERT CHARACTER INTO BUFFER
005360  005261  000000G               INC     LACTIV(R1)      ;SAY ONE MORE PENDING ACTIVATION CHAR
005364  005061  000000G               CLR     LAFSIZ(R1)      ;SAY FIELD WIDTH ACTIVATION NOT IN EFFECT
005370  005061  000000G               CLR     LFWLIM(R1)      ;SAY NO FIELD WIDTH LIMIT
005374  000402                         BR      9$
                                ;
                                ;  User has not typed ahead entire field.
                                ;  Remember field size as activation condition.
                                ;
005376  010061  000000G          1$:    MOV     R0,LAFSIZ(R1)   ;REMEMBER FIELD SIZE
                                ;
                                ;  Finished
                                ;
005402  012604                 9$:     MOV     (SP)+,R4
```

```
     58 005404  012603                          MOV     (SP)+,R3
     59 005406  012602                          MOV     (SP)+,R2
     60 005410  000207                          RETURN
```

```
 1                                  ;-----------------------------------------------------------------------
 2                                  ;  Command - R
 3                                  ;  Turn on high-efficiency tty mode
 4                                  ;
 5 005412  032761  000000G 000000G CMDR:    BIT     #$DETCH,LSW(R1) ;Is this a detached job?
 6 005420  001002                           BNE     9$              ;Br if yes -- Ignore function
 7 005422  004737  005430'                  CALL    HION            ;TURN ON HIGH-EFFICIENCY MODE
 8 005426  000207                  9$:     RETURN
 9                                  ;
10                                  ;  HION IS CALLED TO ENABLE HIGH-EFFICIENCY TTY MODE.
11                                  ;  IT IS CALLED FROM TSEMT AS WELL AS FROM TSEXEC.
12                                  ;  WHEN CALLED, THE LINE INDEX # MUST BE IN R1.
13                                  ;
14 005430  032761  000000G 000000G HION:    BIT     #$DETCH,LSW(R1) ;IS THIS A DETACHED JOB?
15 005436  001003                           BNE     1$              ;BR IF YES (CAN'T USE THIS MODE)
16 005440  052761  000000G 000000G         BIS     #$HITTY,LSW4(R1);ENABLE HIGH-EFFICIENCY MODE
17 005446  000207                  1$:     RETURN
18
19                                  ;-----------------------------------------------------------------------
20                                  ;  Command - S
21                                  ;  Turn on single character activation mode
22                                  ;
23 005450  052761  000000G 000000G CMDS:    BIS     #$CHACT,LSW5(R1);TURN ON SINGLE CHARACTER ACTIVATION
24 005456  000207                           RETURN
25
26                                  ;-----------------------------------------------------------------------
27                                  ;  Command - T
28                                  ;  Turn off single character activation mode
29                                  ;
30 005460  042761  000000G 000000G CMDT:    BIC     #$CHACT,LSW5(R1);TURN OFF SINGLE CHARACTER ACTIVATION
31 005466  000207                           RETURN
32
33                                  ;-----------------------------------------------------------------------
34                                  ;  Command - U
35                                  ;  Enable non-wait TT input  (.TTINR)
36                                  ;
37 005470  052761  000000G 000000G CMDU:    BIS     #$NOWTT,LSW5(R1);ENABLE NON-WAIT TT INPUT
38 005476  000207                           RETURN
```

```
                                ;-----------------------------------------------------------------------
                                ;  Command - V
                                ;  Set field width limit.
                                ;
005500  012761  005510' 000000G CMDV:   MOV     #SFWL,LTSCMD(R1);NEXT CHAR WILL BE FIELD WIDTH SIZE
005506  000207                         RETURN
                                ;
                                ;  This character specifies the field size
                                ;
005510  010346                  SFWL:   MOV     R3,-(SP)
005512  010446                          MOV     R4,-(SP)
005514  010546                          MOV     R5,-(SP)
005516  032761  000000G 000000G        BIT     #$DETCH,LSW(R1) ;Is this a detached job?
005524  001055                         BNE     9$              ;Br if yes -- Ignore function
005526  005061  000000G                CLR     LFWLIM(R1)      ;Initially clear field width
005532  010004                         MOV     R0,R4           ;IS SPECIFIED FIELD WIDTH ZERO?
005534  001451                         BEQ     9$              ;Br if yes
005536  004737  010030'          1$:    CALL    CFTEST          ;IS INPUT COMING FROM A COMMAND FILE?
005542  103046                         BCC     9$              ;BR IF YES
005544  004737  017362'                CALL    SLCHK           ;IS SINGLE LINE EDITOR RUNNING?
005550  103441                         BCS     6$              ;BR IF YES
                                ;
                                ;  See if user typed ahead.
                                ;  Check size of field and discard any chars beyond end of specified size.
                                ;
005552  005003                  14$:    CLR     R3              ;COUNT TOTAL CHARACTERS IN R3
005554  016102  000000G                 MOV     LINPNT(R1),R2   ;GET POINTER TO NEXT CHAR IN BUFFER
005560  020361  000000G          5$:    CMP     R3,LINCNT(R1)   ;HAVE WE CHECKED ALL CHARS IN TT BUFFER?
005564  103033                         BHIS    6$              ;BR IF YES -- NO OVERFLOW HAS OCCURRED YET
005566  010205                         MOV     R2,R5           ;SAVE POINTER TO NEXT CHAR
005570  004737  016420'                CALL    FETCHR          ;GET NEXT CHAR FROM TT INPUT BUFFER
005574  120027  000000G                CMPB    R0,#CR          ;IS THIS CHAR CARRIAGE-RETURN?
005600  001427                         BEQ     9$              ;BR IF YES -- THIS ENDS THE FIELD
005602  032700  000000G                BIT     #ACFLAG,R0      ;IS THIS CHAR AN ACTIVATION CHAR?
005606  001024                         BNE     9$              ;BR IF YES -- FIELD TERMINATED WITHOUT OVRFLOW
005610  120027  000000G                CMPB    R0,#ESCFLG      ;IS THIS CHAR PART OF VT100 ESCAPE SEQ?
005614  001004                         BNE     2$              ;BR IF NOT
005616  032761  000000G 000000G        BIT     #$VTESC,LSW5(R1);ARE WE ACTIVATING ON VT100 ESC SEQUENCES?
005624  001015                         BNE     9$              ;BR IF YES -- FIELD WIDTH OK
005626  005203                   2$:    INC     R3              ;COUNT TOTAL # CHARS IN FIELD
005630  020304                         CMP     R3,R4           ;DOES THIS CHAR OVERFLOW THE FIELD?
005632  101752                         BLOS    5$              ;BR IF NOT
                                ;
                                ;  Field overflow.  Discard all chars typed beyond end of field.
                                ;
005634  010502                         MOV     R5,R2           ;POINT TO CHAR THAT OVERFLOWED FIELD
005636  004737  016712'          10$:   CALL    DELCHR          ;DELETE ALL CHARS THAT ARE BEYOND FIELD END
005642  103375                         BCC     10$             ;LOOP IF MORE CHARS TO DELETE
                                ;  Ring bell to signal user that field overflowed.
005644  012700  000000G                 MOV     #BELL,R0        ;GET BELL CHARACTER
005650  004737  003516'                CALL    QUECHR          ;END TO TERMINAL
                                ;
                                ;  Field did not overflow
                                ;
005654  010461  000000G          6$:    MOV     R4,LFWLIM(R1)   ;SET FIELD WIDTH LIMIT FOR THIS FIELD
                                ;
                                ;  Finished
```

```
      58                                        ;
      59 005660  012605                  9$:       MOV       (SP)+,R5
      60 005662  012604                            MOV       (SP)+,R4
      61 005664  012603                            MOV       (SP)+,R3
      62 005666  000207                            RETURN
```

```
                                ;-----------------------------------------------------------------------------
                                ;  Command - W
                                ;  Turn "paper-tape" mode on.
                                ;
005670  052761  000000G 000000G CMDW:    BIS     #$TAPE,LSW2(R1) ;TURN TAPE MODE ON
005676  000207                          RETURN

                                ;-----------------------------------------------------------------------------
                                ;  Command - X
                                ;  Turn "paper-tape" mode off.
                                ;
005700  042761  000000G 000000G CMDX:    BIC     #$TAPE,LSW2(R1) ;TURN TAPE MODE OFF
005706  000207                          RETURN

                                ;-----------------------------------------------------------------------------
                                ;  Command - Y
                                ;  Disable echoing of line-feed following carriage-return
                                ;
005710  052761  000000G 000000G CMDY:    BIS     #$NOLF,LSW6(R1) ;DISABLE AUTO LINE-FEED ECHO
005716  000207                          RETURN

                                ;-----------------------------------------------------------------------------
                                ;  Command - Z
                                ;  Enable echoing of line-feed following carriage-return
                                ;
005720  042761  000000G 000000G CMDZ:    BIC     #$NOLF,LSW6(R1) ;REENABLE AUTO LINE-FEED ECHO
005726  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.
                                ;
005730  010046                  PCSPND: MOV     R0,-(SP)
005732  004737  017574'                 CALL    SIGWAT          ;SIGNAL JOB WAIT
005736  012700  000000G                 MOV     #S$OTWT,R0      ;CHANGE JOB STATE TO WAITING-FOR-OUTPUT-SPACE
005742  004737  000000G                 CALL    QHDSPN          ;CHANGE STATE AND SUSPEND THE JOB
005746  004737  000000G                 CALL    CHKABT          ;SEE IF WE WERE ABORTED WHILE ASLEEP
005752  012600                          MOV     (SP)+,R0
005754  000207                          RETURN
                                ;-----------------------------------------------------------------------
                                ;  TRYCHR is similar to PUTCHR except if the user's
                                ;  output buffer is full TRYCHR simply returns
                                ;  and does not halt execution.
                                ;  TRYCHR also does not give special treatment to
                                ;  such chars as form feed, backspace, etc.
                                ;  when called r0 must contain the character to be
                                ;  sent and r1 must contain the virtual line index #.
                                ;  all registers are preserved.
                                ;
005756                          TRYCHR: DISABL                  ;** DISABLE **
005764  005761  0000000G              TST     LOTSPC(R1)      ;IS THERE ROOM FOR THE CHAR?
005770  001402                       BEQ     9$              ;BR IF NOT
005772  004737  003516'              CALL    QUECHR          ;PUT CHARACTER INTO BUFFER
005776  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 006000  012700  000000G        OTREGO: MOV     #S$OTFN,R0      ;PUT USER IN OUTPUT-NEEDED QUEUE
      7 006004  004737  000000G                CALL    ENQTL
      8 006010  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.
                                ;
006012  010146                  GETCHR: MOV     R1,-(SP)
006014  010246                          MOV     R2,-(SP)
006016  113701  000000G                 MOVB    CORUSR,R1       ;GET USER'S INDEX #
006022  042761  000000G 000000G         BIC     #$GCESC,LSW3(R1);WE'RE NOT PROCESSING VT50 ESC SEQUENCE
006030  032761  000000C 000000G GCH1:   BIT     #<$DISCN+$CTRLC>,LSW(R1);DISCONNECT OR CTRL-C?
006036  001402                          BEQ     8$              ;BRANCH IF NEITHER
006040  004737  000000G                 CALL    STOP            ;STOP PROGRAM IF EITHER
                                ;
                                ;  See if input is coming from a command file or from the terminal.
                                ;
006044  004737  007332'          8$:     CALL    GTCFCH          ;TRY TO GET A CHARACTER FROM A COMMAND FILE
006050  103402                          BCS     15$             ;BR IF DID NOT GET ONE
006052  000137  006412'                  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.
                                ;
006056  032761  000000G 000000G 15$:     BIT     #$LOFCF,LSW9(R1);Are we processing a logoff command file?
006064  001405                          BEQ     13$             ;Br if not
006066  052761  000000G 000000G         BIS     #$DOOFF,LSW(R1) ;Force job logoff
006074  004737  000000G                 CALL    STOP
                                ;
                                ;  If this is a detached job, halt its execution if command file finished
                                ;
006100  032761  000000G 000000G 13$:     BIT     #$DETCH,LSW(R1) ;Is this a detached job?
006106  001405                          BEQ     16$             ;Br if not
006110  052761  000000G 000000G         BIS     #$DISCN,LSW(R1) ;Force job logoff
006116  004737  000000G                 CALL    STOP            ;Halt detached jobs that want terminal input
                                ;
                                ;  See if we should get a character from the single line editor.
                                ;
006122  032761  000000C 000000G 16$:     BIT     #<SPCTTY!DISSLE>,LJSW(R1);Special TTY mode or SL disabled?
006130  001031                          BNE     11$             ;Br if yes -- don't use SL
006132  032761  000000G 000000G         BIT     #$SLON,LSW7(R1) ;Is SL enabled for this line?
006140  001425                          BEQ     11$             ;Br if not
006142  032761  000000C 000000G         BIT     #<$ODTMD!$HITTY>,LSW4(R1);Are we in ODT or high efficiency?
006150  001021                          BNE     11$             ;Br if yes
006152  032761  000000G 000000G         BIT     #$VTESC,LSW5(R1);VTxxx activation enabled?
006160  001015                          BNE     11$             ;Br if yes
006162  032761  000000G 000000G         BIT     #$GTLIN,LSW4(R1);Is a .GTLIN being done?
006170  001004                          BNE     12$             ;Br if yes
006172  032761  000000G 000000G         BIT     #$SLTTY,LSW7(R1);Is SL enabled for TTY input?
006200  001405                          BEQ     11$             ;Br if not
006202                           12$:    OCALL   GTSLCH          ;Get character from single line editor
006210  000137  007044'                  JMP     GCEXIT          ;Exit from GETCHR
                                ;
                                ;  Get character from the terminal.
                                ;
```

```
     58 006214  032761  000000G 000000G 11$:     BIT     #$NOINT,LSW7(R1);Does user want non-interactive execution?
     59 006222  001006                          BNE     14$             ;Br if yes
     60 006224  013761  000000G 000000G         MOV     VINTIO,LHIPCT(R1);Reset interactive I/O limit for job
     61 006232  013761  000000G 000000G         MOV     VQUAN1,LITIME(R1);Reset interactive CPU time limit
     62 006240  005761  000000G         14$:    TST     LACTIV(R1)      ;Any activation chars pending?
     63 006244  001003                          BNE     9$              ;Branch if yes
     64 006246  004737  011240'                CALL    ILWAIT          ;Wait for activation
     65 006252  000666                          BR      GCH1            ;Go try again
     66 006254  016102  000000G         9$:     MOV     LINPNT(R1),R2   ;Get pointer to next char to get
     67 006260  004737  016712'                CALL    DELCHR          ;Get char and delete from input buffer
     68 006264  103661                          BCS     GCH1            ;Br if no more characters are available
     69 006266  005061  000000G                 CLR     LRTCHR(R1)      ;Say no read time-out pending
     70                                 ;
     71                                 ;  See if we are in single-character-activation mode.
     72                                 ;
     73 006272  032761  000000G 000000G         BIT     #SPCTTY,LJSW(R1);DOES PROGRAM WANT SINGLE-CHAR INPUT?
     74 006300  001415                          BEQ     6$              ;BR IF NOT
     75 006302  032761  000000G 000000G         BIT     #$CHACT,LSW5(R1);IS SINGLE-CHARACTER INPUT ENABLED?
     76 006310  001411                          BEQ     6$              ;BR IF NOT
     77 006312  004737  017522'                CALL    CVTLC           ;SET IF WE NEED TO CONVERT TO UPPER-CASE
     78 006316  032700  000000G                 BIT     #ACFLAG,R0      ;IS THIS AN ACTIVATION CHARACTER?
     79 006322  001402                          BEQ     10$             ;BR IF NOT
     80 006324  000137  006562'                 JMP     GCCKDS
     81 006330  000137  006626'         10$:    JMP     GCCKCC
     82                                 ;
     83                                 ;  See if we are in high-efficiency mode.
     84                                 ;
     85 006334  032761  000000G 000000G 6$:      BIT     #$HITTY,LSW4(R1);ARE WE IN HIGH-EFFICIENCY MODE?
     86 006342  001402                          BEQ     3$              ;BR IF NOT
     87                                 ;
     88                                 ;  We are in high-efficiency mode
     89                                 ;
     90 006344  000137  007002'         5$:     JMP     GCEND           ;DO EXPRESS CHARACTER PROCESSING
     91                                 ;
     92                                 ;  SEE IF THIS IS FLAG CHAR MEANING NEXT CHAR IS PART OF
     93                                 ;  VT50 ESCAPE SEQUENCE.
     94                                 ;
     95 006350  032761  000000G 000000G 3$:      BIT     #$GCESC,LSW3(R1);IS THIS CHAR PART OF ESC SEQUENCE?
     96 006356  001402                          BEQ     1$              ;BR IF NOT
     97 006360  000137  006546'                 JMP     GCCKAC          ;TAKE CHAR WITHOUT FURTHER CHECKING
     98 006364  120027  000000G         1$:     CMPB    R0,#ESCFLG      ;IS THIS FLAG FOR NEXT CHAR IN ESC SEQ?
     99 006370  001010                          BNE     GCCKES          ;BR IF NOT
    100 006372  032761  000000G 000000G         BIT     #$VTESC,LSW5(R1);ARE WE ACTIVATING ON ESCAPE SEQUENCES?
    101 006400  001404                          BEQ     GCCKES          ;BR IF NOT
    102 006402  052761  000000G 000000G         BIS     #$GCESC,LSW3(R1);REMEMBER NEXT CHAR PART OF ESC SEQ
    103 006410  000721                          BR      9$              ;GO GET NEXT CHAR
```

```
 1                                      ;
 2                                      ;  WE HAVE THE NEXT CHARACTER IN R0.
 3                                      ;  SEE IF WE NEED TO TRANSLATE LOWER CASE CHARS TO UPPER CASE.
 4                                      ;
 5 006412  004737  017522'      GCCKES: CALL    CVTLC           ;CONVERT LC TO UC IF NEEDED
 6                                      ;
 7                                      ;  SEE IF WE NEED TO DO ECHO SUPPRESSION.
 8                                      ;
 9 006416  032761  000000G 000000G      BIT     #SPCTTY,LJSW(R1);IS ECHO SUPPRESSION NEEDED?
10 006424  001450                      BEQ     GCCKAC          ;BRANCH IF NOT
11 006426  032761  000000G 000000G      BIT     #$CHACT,LSW5(R1);ARE WE IN SINGLE-CHAR-ACTIVATION MODE?
12 006434  001074                      BNE     GCCKCC          ;BR IF YES
13                                      ;
14                                      ;  BEGIN ECHO SUPPRESSION.
15                                      ;
16 006436  010346                      MOV     R3,-(SP)
17 006440  010002                      MOV     R0,R2           ;GET THE CHAR
18 006442  042702  177400              BIC     #177400,R2      ;CLEAR ACTIVATION CHAR FLAG
19                                      ;  SEE IF THIS IS A REGULAR OR CONTROL CHARACTER
20 006446  120227  000037              CMPB    R2,#37          ;REGULAR OR CONTROL?
21 006452  101013                      BHI     4$              ;BRANCH IF REGULAR CHAR
22                                      ;  THIS IS A CONTROL CHAR -- CHECK FOR SPECIAL CASES.
23 006454  012703  000004              MOV     #NESCTL,R3      ;# OF SPECIAL CONTROL CHARS
24 006460  120263  007056'      5$:     CMPB    R2,SESCTL(R3)   ;IS THIS ONE?
25 006464  001402                      BEQ     8$              ;BRANCH IF YES
26 006466  005303                      DEC     R3              ;TRY REST
27 006470  100373                      BPL     5$
28                                      ;  FALL THROUGH WITH R3=-1 FOR REGULAR CONTROL CHAR.
29 006472  006303              8$:     ASL     R3              ;GET WORD TABLE INDEX
30 006474  016303  007066'              MOV     SESRTN(R3),R3   ;GET ROUTINE ADDRESS
31 006500  000402                      BR      6$
32                                      ;  THIS IS NOT A CONTROL CHAR.
33 006502  012703  004112'      4$:     MOV     #ESUAC,R3       ;SET NORMAL HANDLER ROUTINE
34 006506  010361  000000G      6$:     MOV     R3,LESRTN(R1)   ;SET HANDLER ROUTINE FOR PUTCHR
35 006512  120227  000141              CMPB    R2,#141         ;IS CHAR A LOWER-CASE LETTER?
36 006516  103405                      BLO     1$              ;BR IF NOT
37 006520  120227  000172              CMPB    R2,#172
38 006524  101002                      BHI     1$
39 006526  162702  000040              SUB     #40,R2          ;CONVERT LOWER-CASE TO UPPER-CASE
40 006532  110261  000000G      1$:     MOVB    R2,LESCHR(R1)   ;REMEMBER CHAR
41 006536  052761  000000G 000000G      BIS     #$NOOUT,LSW3(R1);TURN ON ECHO SUPPRESSION
42 006544  012603                      MOV     (SP)+,R3
```

```
                                        ;
                                        ;  See if this is an activation character.
                                        ;
006546  004737  010030'         GCCKAC: CALL    CFTEST          ;INPUT FROM CONTROL FILE?
006552  103017                          BCC     GCCKCE          ;BR IF YES
006554  032700  000000G                 BIT     #ACFLAG,R0      ;IS THIS AN ACTIVATION CHAR?
006560  001502                          BEQ     GCCKDE          ;BRANCH IF NOT.
                                        ;
                                        ;  This is an activation character.
                                        ;  See if we need to start doing deferred echoing.
                                        ;
006562  032761  000000G 000000G GCCKDS: BIT     #$DODFR,LSW3(R1);IS ECHOING BEING DEFERRED?
006570  001416                          BEQ     GCCKCC          ;BRANCH IF NOT
006572  032761  000000G 000000G         BIT     #$GCECO,LSW3(R1);HAVE WE ALREADY STARTED?
006600  001010                          BNE     GCEAC           ;BRANCH IF YES
006602  052761  000000G 000000G         BIS     #$GCECO,LSW3(R1);BEGIN DEFERRED ECHOING WITH NEXT CHAR
006610  000406                          BR      GCCKCC
                                        ;
                                        ;  See if want to list the contents of a control file.
                                        ;
006612  032761  000000G 000000G GCCKCE: BIT     #$QUIET,LSW4(R1);LIST COMMAND FILE?
006620  001002                          BNE     GCCKCC          ;BR IF NO LIST
                                        ;
                                        ;  ECHO THE ACTIVATION CHAR NOW.
                                        ;
006622  004737  007100'         GCEAC:  CALL    GCECHO          ;ECHO THE CHARACTER
                                        ;
                                        ;  See if character we got is control-c.
                                        ;
006626  120027  000000G         GCCKCC: CMPB    R0,#CTRLC       ;IS CHAR CTRL-C?
006632  001063                          BNE     GCEND           ;BRANCH IF NOT
006634  032761  000000G 000000G         BIT     #$DBGMD,LSW6(R1);IS A DEBUGGING PROGRAM IN CONTROL?
006642  001024                          BNE     7$              ;BR IF YES
006644  032761  000000G 000000G         BIT     #$UCTLC,LSW4(R1);IS CTRL-C A USER DEF ACTIV CHAR?
006652  001053                          BNE     GCEND           ;BRANCH IF IT IS
006654  005761  000000G                 TST     LSCCA(R1)       ;DID USER DO .SCCA?
006660  001050                          BNE     GCEND           ;BR IF YES
006662  032761  000000G 000000G         BIT     #$SCCA,LSW5(R1) ;Suppressing control-C aborts for program?
006670  001402                          BEQ     5$              ;Br if no
006672  000137  006030'                 JMP     GCH1            ;If yes, throw away ^C, and get next char.
006676  032761  000000G 000000G 5$:     BIT     #$CCLRN,LSW5(R1);IS CCL TRANSLATOR RUNNING?
006704  001403                          BEQ     7$              ;BR IF NOT
006706  004737  010030'                 CALL    CFTEST          ;IS INPUT COMING FROM A COMMAND FILE?
006712  103033                          BCC     GCEND           ;BR IF YES
006714  032737  000000G 000000G 7$:     BIT     #LF$IN,LOGFLG   ;Are we logging input characters?
006722  001417                          BEQ     8$              ;BR IF NOT
006724  004737  010030'                 CALL    CFTEST          ;IS INPUT COMING FROM A CONTROL FILE?
006730  103404                          BCS     6$              ;BR IF NOT
006732  032761  000000G 000000G         BIT     #$QUIET,LSW4(R1);SHOULD WE DISPLAY CONTROL FILES?
006740  001010                          BNE     8$              ;BR IF NOT
006742  032761  000000G 000000G 6$:     BIT     #$ECHO,LSW2(R1) ;ARE WE ECHOING CHARACTERS?
006750  001404                          BEQ     8$              ;DON'T LOG IF NOT ECHOING
006752  004737  010776'                 CALL    LOGCHR          ;LOG THE CONTROL-C
006756  004737  011046'                 CALL    LOGCR           ;LOG CR-LF
006762  004737  000000G         8$:     CALL    STOP            ;STOP PROGRAM EXECUTION
```

```
                                        ;
                                        ;   THIS CHAR IS NOT AN ACTIVATION CHAR.
                                        ;
                                        ;   SEE IF WE NEED TO DO DEFERRED ECHOING.
                                        ;
006766  032761  000000G 000000G GCCKDE:  BIT     #$GCECO,LSW3(R1);HAVE WE STARTED DEFERRED ECHOING?
006774  001402                          BEQ     GCEND           ;BRANCH IF NOT
006776  004737  007100'                  CALL    GCECHO          ;ECHO THE CHARACTER
                                        ;
                                        ;   See if we should write character to log file
                                        ;
007002  032737  000000G 000000G GCEND:   BIT     #LF$IN,LOGFLG   ;Are we logging input characters?
007010  001415                          BEQ     GCEXIT          ;Br if not
007012  004737  010030'                  CALL    CFTEST          ;Is input coming from a control file?
007016  103404                          BCS     1$              ;Br if not
007020  032761  000000G 000000G         BIT     #$QUIET,LSW4(R1);Should we log control file characters?
007026  001006                          BNE     GCEXIT          ;Br if not
007030  032761  000000G 000000G 1$:     BIT     #$ECHO,LSW2(R1) ;Is echo suppression in effect?
007036  001402                          BEQ     GCEXIT          ;Do not log if not echoing
007040  004737  010776'                  CALL    LOGCHR          ;Log the character
007044  042700  177400          GCEXIT: BIC     #^C377,R0       ;Mask character to 8 bits
007050  012602                          MOV     (SP)+,R2
007052  012601                          MOV     (SP)+,R1
007054  000207                          RETURN
```

```
      1                                 ;
      2                                 ;   TABLE OF CONTROL CHARACTERS WHICH REQUIRE SPECIAL
      3                                 ;   PROCESSING WITH REGARD TO ECHO SUPPRESSION.
      4                                 ;
      5                                 ;   TABLE OF CONTROL CHARACTERS.
      6                                 ;
      7 007056      000G                SESCTL: .BYTE   FF              ;FORM FEED
      8 007057      000G                        .BYTE   ESC             ;ESCAPE
      9 007060      000G                        .BYTE   LF              ;LINE FEED
     10 007061      000G                        .BYTE   CR              ;CARRIAGE RETURN
     11 007062      000G                        .BYTE   TAB             ;TAB
     12           000004                NESCTL  =       .-SESCTL-1      ;# OF SPECIAL CHARS.
     13                                         .EVEN
     14                                 ;
     15                                 ;   PARALLEL TABLE OF ADDRESSES OF ROUTINES TO RESTART
     16                                 ;   OUTPUT WHEN ECHO SUPPRESSION IS IN EFFECT FOR
     17                                 ;   SPECIAL CONTROL CHAR.
     18                                 ;   SESRTN TABLE MUST BE PARALLEL TO SESCTL TABLE.
     19                                 ;   NOTE:  (-1) TABLE ENTRY IS USED FOR REGULAR CONTROL CHARS.
     20                                 ;
     21 007064  004144'                         .WORD   ESCTL           ;(-1) REGULAR CONTROL CHAR
     22 007066  004210'                 SESRTN: .WORD   ESFF            ;FORM FEED
     23 007070  004226'                         .WORD   ESESC           ;ESCAPE
     24 007072  004172'                         .WORD   ESCRLF          ;LINE FEED
     25 007074  004172'                         .WORD   ESCRLF          ;CARRIAGE RETURN
     26 007076  004236'                         .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.
                                ;
007100  004737  017322'       GCECHO: CALL    SCACHK          ;ARE WE IN SINGLE CHARACTER INPUT MODE?
007104  103511                        BCS     99$             ;BR IF YES -- DON'T ECHO CHARACTER
007106  032761  000000G 000000G       BIT     #$ECHO,LSW2(R1) ;IS CHAR ECHOING WANTED?
007114  001505                        BEQ     99$             ;RETURN IF NOT
007116  032761  000000G 000000G       BIT     #$GCESC,LSW3(R1);IS THIS CHAR PART OF ESC SEQUENCE?
007124  001101                        BNE     99$             ;BR IF YES
007126  010046                        MOV     R0,-(SP)
007130  010246                        MOV     R2,-(SP)
007132  010346                        MOV     R3,-(SP)
007134  042700  000000G               BIC     #ACFLAG,R0      ;STRIP OFF ACTIVATION-CHAR FLAG
                                ;
                                ;  See if we should echo line-feed characters
                                ;
007140  120027  000000G               CMPB    R0,#LF          ;IS THIS CHAR LINE-FEED?
007144  001011                        BNE     7$              ;BR IF NOT
007146  032761  000000G 000000G       BIT     #$NOLF,LSW6(R1) ;IS LINE-FEED ECHO SUPPRESSION IN EFFECT?
007154  001405                        BEQ     7$              ;BR IF NOT
007156  032761  000000G 000000G       BIT     #$DBGMD,LSW6(R1);IS A DEBUGGER RUNNING NOW?
007164  001456                        BEQ     2$              ;BR IF NOT (DON'T ECHO LF)
007166  000453                        BR      5$              ;ECHO LF
                                ;
                                ;  SEE IF CHAR IS A USER DEFINED ACTIVATION CHAR.
                                ;
007170  016102  000000G         7$:   MOV     LNSPAC(R1),R2   ;GET # OF USER DEF ACTIV CHARS
007174  001405                        BEQ     1$              ;BRANCH IF NONE
007176  016103  000000G               MOV     LSPACT(R1),R3   ;POINT TO TABLE FOR USER
007202  120023                 3$:   CMPB    R0,(R3)+        ;SEE IF THIS IS ONE
007204  001446                        BEQ     2$              ;BRANCH IF IT IS
007206  077203                        SOB     R2,3$           ;LOOP IF MORE TO CHECK
                                ;  SEE IF CHARACTER IS ESCAPE.
007210  120027  000037         1$:   CMPB    R0,#37          ;REGULAR OR CONTROL CHAR?
007214  101040                        BHI     5$              ;BRANCH IF REGULAR CHARACTER
007216  120027  000000G               CMPB    R0,#ESC         ;IS CHAR ESCAPE?
007222  001003                        BNE     4$              ;BRANCH IF NOT
007224  112700  000044               MOVB    #'$,R0          ;OTHERWISE, ECHO $ FOR ESCAPE
007230  000432                        BR      5$
                                ;  CHECK FOR SPECIAL CONTROL CHARACTERS.
007232  120027  000000G         4$:   CMPB    R0,#CTRLZ       ;CTRL-Z?
007236  001406                        BEQ     6$              ;BRANCH IF YES
007240  120027  000000G               CMPB    R0,#CTRLC       ;CTRL-C?
007244  001403                        BEQ     6$
007246  120027  000000G               CMPB    R0,#CTRLX       ;CTRL-X?
007252  001021                        BNE     5$              ;BRANCH IF NOT SPECIAL CONTROL CHAR
                                ;  ECHO ^-CHAR FOR SPECIAL CONTROL CHARS
007254  110003                 6$:   MOVB    R0,R3           ;SAVE CONTROL CHAR
007256  112700  000136               MOVB    #136,R0         ;ECHO ^
007262  004737  003112'              CALL    PUTCH1
007266  110300                        MOVB    R3,R0           ;GET CONTROL CHAR
007270  052700  000100               BIS     #100,R0         ;CONVERT TO PRINTING CHAR
007274  004737  003112'              CALL    PUTCH1          ;PRINT CHAR
```

```
     58 007300  112700  000000G                 MOVB    #CR,R0          ;ECHO CR
     59 007304  004737  003112'                 CALL    PUTCH1
     60 007310  112700  000000G                 MOVB    #LF,R0
     61 007314  000400                          BR      5$
     62                                 ;
     63                                 ;  THIS IS A REGULAR CHARACTER.
     64                                 ;
     65                                 ;  ECHO CHAR IN R0.
     66 007316  004737  003112'         5$:     CALL    PUTCH1          ;ECHO THE CHARACTER
     67                                 ;  RETURN
     68 007322  012603                  2$:     MOV     (SP)+,R3
     69 007324  012602                          MOV     (SP)+,R2
     70 007326  012600                          MOV     (SP)+,R0
     71 007330  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.
                                ;
007332  004737  010030'       GTCFCH: CALL    CFTEST          ;SEE IF INPUT IS COMING FROM A COMMAND FILE
007336  103543                        BCS     13$             ;BR IF NOT
                                ;
                                ;  See if we are holding a character
                                ;
007340  113700  000000G               MOVB    CFHOLD,R0       ;ARE WE HOLDING A CHARACTER?
007344  001403                        BEQ     10$             ;BR IF NOT
007346  105037  000000G               CLRB    CFHOLD          ;SAY CHARACTER IS GONE
007352  000532                        BR      12$             ;TAKE SUCCESSFUL RETURN
                                ;
                                ;  Input is coming from a command file.
                                ;
007354  004737  007650'       10$:    CALL    CFCHAR          ;GET CHAR FROM CONTROL FILE
007360  103531                        BCS     14$             ;BR IF END OF COMMAND FILE HIT
007362  120027  000136                CMPB    R0,#'^          ;IS THIS A CONTROL CHAR?
007366  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.
                                ;
007370  004737  007650'               CALL    CFCHAR          ;GET NEXT CHAR
007374  103523                        BCS     14$             ;BR IF END OF COMMAND FILE HIT
                                ;
                                ;  ^$ is interpreted as escape
                                ;
007376  120027  000044                CMPB    R0,#'$          ;TREAT ^$ AS ESCAPE CHAR
007402  001003                        BNE     3$
007404  012700  000000G               MOV     #ESC,R0
007410  000513                        BR      12$
                                ;
                                ;  ^^ is interpreted as a single ^
                                ;
007412  120027  000136          3$:   CMPB    R0,#'^          ;ANOTHER "^"?
007416  001510                        BEQ     12$             ;IF YES THEN RETURN "^" AS CHARACTER
                                ;
                                ;  ^( means stop listing control file
                                ;
007420  120027  000050                CMPB    R0,#'(          ;STOP LISTING COMMAND?
007424  001007                        BNE     4$              ;BR IF NOT
007426  052761  000000G 000000G       BIS     #$QUIET,LSW4(R1);SET NO LISTING FLAG
007434  042761  000000G 000000G       BIC     #$CFSOT,LSW4(R1);ALLOW PROGRAM OUTPUT TO PRINT
007442  000744                        BR      10$             ;GO GET NEXT CHAR FROM FILE
                                ;
                                ;  ^) means start listing control file
```

```
                                ;
007444  120027  000051          4$:     CMPB    R0,#')          ;START-LISTING COMMAND?
007450  001004                          BNE     7$              ;BR IF NOT
007452  042761  000000C 000000G        BIC     #<$QUIET!$CFSOT>,LSW4(R1);START LISTING FILE
007460  000735                          BR      10$             ;GO GET NEXT CHAR
                                ;
                                ;  ^! means suppress all output -- command file and program
                                ;
007462  120027  000041          7$:     CMPB    R0,#'!          ;SUPPRESS OUTPUT?
007466  001004                          BNE     5$              ;BR IF NOT
007470  052761  000000C 000000G        BIS     #<$CFSOT!$QUIET>,LSW4(R1);BEGIN SUPPRESSING OUTPUT
007476  000726                          BR      10$
                                ;
                                ;  ^> means accept all chars from @file
                                ;
007500  120027  000076          5$:     CMPB    R0,#'>          ;ACCEPT ALL CHARS?
007504  001004                          BNE     6$              ;BR IF NOT
007506  052761  000000G 000000G        BIS     #$CFALL,LSW4(R1);SET FLAG
007514  000717                          BR      10$
                                ;
                                ;  ^< means accept only .gtlin chars from @file
                                ;
007516  120027  000074          6$:     CMPB    R0,#'<          ;ACCEPT ONLY .GTLIN CHARS?
007522  001004                          BNE     2$              ;BR IF NOT
007524  042761  000000G 000000G        BIC     #$CFALL,LSW4(R1);RESET FLAG
007532  000677                          BR      GTCFCH
                                ;
                                ;  ^digit is used to indicate a parameter substitution
                                ;
007534  120027  000061          2$:     CMPB    R0,#'1          ;^DIGIT MEANS SUBSTITUTE PARAM
007540  103414                          BLO     1$              ;BR IF NOT PARAM
007542  120027  000060G               CMPB    R0,#<60+MXCPRM> ;IN VALID RANGE?
007546  101011                          BHI     1$              ;BR IF NOT
                                ;
                                ;  Switch input to a parameter string
                                ;
007550  042700  177760                  BIC     #177760,R0      ;LEAVE VALUE ONLY
007554  006300                          ASL     R0              ;CONVERT TO WORD TABLE INDEX
007556  016037  177776G 000000G        MOV     <PRMPNT-2>(R0),CURPRM ;SET CHAR STRING POINTER
007564  004737  000000G               CALL    CHKABT          ;ALLOW ABORTS WHILE GETTING PARAMETERS
007570  000671                          BR      10$             ;GO GET 1ST CHAR FROM STRING
                                ;
                                ;  ^letter causes the letter to be converted to a control character
                                ;
007572  120027  000101          1$:     CMPB    R0,#'A          ;IS THIS A LETTER?
007576  103414                          BLO     15$             ;BR IF NOT
007600  120027  000132                  CMPB    R0,#'Z
007604  101406                          BLOS    16$             ;BR IF UPPER-CASE LETTER
007606  120027  000141                  CMPB    R0,#141         ;SEE IF THIS IS A LOWER-CASE LETTER
007612  103406                          BLO     15$             ;BR IF NOT
007614  120027  000172                  CMPB    R0,#172
007620  101003                          BHI     15$             ;BR IF NOT LETTER
007622  042700  177740          16$:    BIC     #177740,R0      ;CONVERT ALPHA TO CONTROL CHAR
007626  000404                          BR      12$             ;RETURN CONTROL CHARACTER AS CHARACTER GOTTEN
                                ;
                                ;  ^ was followed by something we don't recognize.
                                ;  Just return the ^ as the character gotten.
```

```
    115                                         ;
    116 007630  110037  000006          15$:    MOVB    R0,CFHOLD       ;"PUSH" THE NEXT CHARACTER
    117 007634  112700  000136                  MOVB    #'^,R0          ;RETURN "^" AS THIS CHARACTER
    118                                         ;
    119                                         ;  Finished
    120                                         ;
    121 007640  000241                  12$:    CLC                     ;INDICATE SUCCESSFUL RETURN
    122 007642  000401                          BR      13$
    123 007644  000261                  14$:    SEC                     ;INDICATE UNSUCCESSFUL RETURN
    124 007646  000207                  13$:    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.
                                ;
007650  010246                  CFCHAR: MOV     R2,-(SP)
007652  013702  000000G               MOV     CURPRM,R2       ;INPUT FROM PARAM STRING?
007656  001407                       BEQ     5$              ;BR IF NOT
                                ;
                                ;  INPUT IS COMING FROM A PARAMETER STRING
                                ;
007660  112200                       MOVB    (R2)+,R0        ;GET NEXT CHAR FROM STRING
007662  001403                       BEQ     6$              ;BR IF HIT END OF STRING
007664  010237  000000G               MOV     R2,CURPRM       ;UPDATE CHAR POINTER
007670  000452                       BR      9$              ;RETURN
                                ;  HIT END OF PARAMETER STRING.
                                ;  SWITCH INPUT BACK TO CONTROL FILE.
007672  005037  000000G         6$:   CLR     CURPRM          ;CLEAR PARAM STRING POINTER
                                ;
                                ;  GET CHARACTER FROM CONTROL FILE
                                ;
007676  013702  000000G         5$:   MOV     CFPNT,R2        ;GET POINTER INTO BUFFER
007702  020227  000000G         4$:   CMP     R2,#CFEND       ;HIT END OF BUFFER?
007706  103437                       BLO     1$              ;BRANCH IF NOT
                                ;  REACHED END OF BUFFER -- READ IN NEXT BLOCK.
007710  113746  000052               MOVB    @#52,-(SP)      ;SAVE I/O ERROR CODE
007714  005237  000000G               INC     CFBLK           ;INC FILE BLOCK NUMBER
007720                               .READW  #CFARG,#CFCHAN,#CFBUF,#256.,CFBLK
007760  112637  000052               MOVB    (SP)+,@#52      ;REPLACE ERROR CELL
007764  103006                       BCC     3$              ;BR IF NOT AT END OF FILE
                                ;
                                ;  End of file has been hit.
                                ;  Try to pop up to higher level file.
                                ;
007766  004737  010152'               CALL    CFPOP           ;TRY TO POP UP TO HIGHER LEVEL FILE
007772  005737  000000G               TST     CFPNT           ;WAS THERE A HIGHER LEVEL FILE?
007776  001337                       BNE     5$              ;BR IF YES
010000  000410                       BR      11$
                                ;  GET NEXT CHAR FROM BUFFER
010002  012702  000000G         3$:   MOV     #CFBUF,R2       ;RESET BUFFER POINTER
010006  112200                1$:   MOVB    (R2)+,R0        ;GET NEXT CHAR
010010  001734                       BEQ     4$              ;IGNORE NULLS
010012  010237  000000G         8$:   MOV     R2,CFPNT        ;SAVE NEW CHAR POINTER
                                ;  Successful return.
010016  000241                  9$:   CLC
010020  000401                       BR      12$
                                ;  Unsuccessful return.
010022  000261                  11$:  SEC                     ;SIGNAL UNSUCCESSFUL RETURN
```

```
     58 010024  012602                    12$:     MOV      (SP)+,R2
     59 010026  000207                             RETURN
```

```
      1                                          .SBTTL  CFTEST -- Determine if TT input is from file
      2                                ;-----------------------------------------------------------
      3                                ;  CFTEST IS CALLED TO DETERMINE IF TT INPUT IS COMING FROM
      4                                ;  A COMMAND FILE.
      5                                ;  IF YES, THE C-FLAG IS RESET ON RETURN.
      6                                ;  IF NO, THE C-FLAG IS SET ON RETURN.
      7                                ;  WHEN CALLED, R1 MUST CONTAIN THE USER INDEX NUMBER.
      8                                ;  ALL REGISTERS ARE PRESERVED.
      9                                ;
     10 010030  005737  000000G        CFTEST: TST     CFPNT           ;INPUT FROM COMMAND FILE?
     11 010034  001417                         BEQ     CFTNO           ;BR IF NOT
     12 010036  000404                         BR      CFTST1          ;NOP this br to disable debug cmd file input
     13 010040  032761  000000G 000000G       BIT     #$DBGMD,LSW6(R1) ;IS DEBUGGER IN CONTROL?
     14 010046  001012                         BNE     CFTNO           ;BR IF SO - CAN'T DEBUG FROM CMD FILE!
     15 010050  032761  000000G 000000G CFTST1: BIT     #$GTLIN,LSW4(R1);IS THIS A .GTLIN INPUT EMT?
     16 010056  001004                         BNE     2$              ;BR IF YES
     17 010060  032761  000000G 000000G       BIT     #$CFALL,LSW4(R1);GET .TTYIN INPUT FROM @FILE?
     18 010066  001402                         BEQ     CFTNO           ;BR IF NOT
     19 010070  000241                  2$:    CLC                     ;SAY INPUT COMING FROM FILE
     20 010072  000207                         RETURN
     21 010074  000261                  CFTNO: SEC                     ;SAY INPUT NOT COMING FROM FILE
     22 010076  000207                         RETURN
```

```
                                        .SBTTL  CFSTOP -- Suspend command file input
                                ;-----------------------------------------------------------------------
                                ;  Suspend command file input by setting CFPNT=0.
                                ;  All registers are preserved.
                                ;
010100  010046                 CFSTOP: MOV     R0,-(SP)
                                ;
                                ;  Say input not coming from a command file
                                ;
010102  005037  000000G                 CLR     CFPNT           ;Suspend command file input
                                ;
                                ;  Clear command-file-active status flag in RMON cell
                                ;
010106  013700  000000G                 MOV     CXTRMN,R0       ;Get virtual address of RMON in cxt blk
010112  042760  000000G 000000G         BIC     #CFACFL,R$CFST(R0) ;Say command file not active
                                ;
                                ;  Finished
                                ;
010120  012600                          MOV     (SP)+,R0
010122  000207                          RETURN

                                        .SBTTL  CFSTRT -- Restart command file input
                                ;-----------------------------------------------------------------------
                                ;  Restart command file input by storing a non-zero value into CFPNT.
                                ;
                                ;  Inputs:
                                ;   R0 = Value to store into CFPNT
                                ;
010124  010046                  CFSTRT: MOV     R0,-(SP)
                                ;
                                ;  Restart command file input
                                ;
010126  010037  000000G                 MOV     R0,CFPNT        ;Set command file buffer pointer
010132  001405                          BEQ     9$              ;Br if not starting command file
                                ;
                                ;  Set command-file-active flag in RMON status cell
                                ;
010134  013700  000000G                 MOV     CXTRMN,R0       ;Get virtual address of RMON in cxt blk
010140  052760  000000G 000000G         BIS     #CFACFL,R$CFST(R0) ;Set command-file-active flags
                                ;
                                ;  Finished
                                ;
010146  012600                  9$:     MOV     (SP)+,R0
010150  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.
                                ;
010152  010146                 CFPOP:  MOV     R1,-(SP)
010154  010346                         MOV     R3,-(SP)
010156  010446                         MOV     R4,-(SP)
010160  010546                         MOV     R5,-(SP)
010162  113746  000052                 MOVB    @#52,-(SP)      ;SAVE I/O ERROR CODE CELL
010166  113701  000000G                MOVB    CORUSR,R1       ;GET JOB INDEX NUMBER
010172  042761  000000G 000000G        BIC     #$CFCCL,LSW4(R1);SAY WE ARE NOT GETTING CHARS FROM CCL COMMAND
010200  005737  000000G                TST     CFPNT           ;IS A COMMAND FILE IN USE NOW?
010204  001002                         BNE     11$             ;Br if yes
010206  000137  010760'                JMP     9$
010212  013705  000000G        11$:    MOV     CXTRMN,R5       ;GET ADDRESS OF SIMULATED RMON DATA
010216  113765  000000G 000000G        MOVB    CFIND,R$INST(R5);RESTORE IND STATUS BYTE
010224  105037  000000G                CLRB    CFHOLD          ;CLEAR ANY COMMAND FILE HOLDING CHAR
                                ;
                                ;  Close currently open file
                                ;
010230  032761  000000G 000000G        BIT     #$CFOPN,LSW4(R1);IS THE COMMAND FILE CHANNEL OPEN?
010236  001406                         BEQ     1$              ;BR IF NOT
010240                                 .PURGE  #CFCHAN         ;CLOSE CURRENT FILE
010246  042761  000000G 000000G        BIC     #$CFOPN,LSW4(R1);SAY COMMAND FILE CHANNEL IS NOW CLOSED
                                ;
                                ;  See if there is a higher level command file to restore
                                ;
010254  105737  000000G        1$:     TSTB    CFNEST          ;ANY HIGHER LEVEL COMMAND FILES?
010260  001060                         BNE     2$              ;BR IF YES
                                ;
                                ;  There are no higher level command files
                                ;
010262  004737  010100'                 CALL    CFSTOP          ;Say no data coming from command file
010266  042761  000000C 000000G        BIC     #<$CFALL!$CFSOT>,LSW4(R1) ;CLEAR COMMAND FILE CONTROL FLAGS
010274  032761  000000G 000000G        BIT     #$LOFCF,LSW9(R1);Are we leaving a log off command file?
010302  001006                         BNE     15$             ;If so, do not restore terminal input
010304  042761  000000G 000000G        BIC     #$NOIN,LSW3(R1) ;ALLOW INPUT TO BE ACCEPTED FOR LINE
010312  042761  000000G 000000G        BIC     #$SUCF,LSW9(R1) ;Say we are no longer executing startup file
010320  005000                 15$:    CLR     R0              ;Init privilege vector index
010322  016060  000000G 000000G 10$:    MOV     PRIVS0(R0),PRIVF0(R0);Reset command file priv to set priv
010330  032761  000000G 000000G        BIT     #$INKMN,LSW4(R1);Are we in TSKMON now?
010336  001403                         BEQ     12$             ;Br if not
010340  016060  000000G 000000G        MOV     PRIVS0(R0),PRIVC0(R0) ;Reset current privileges
010346  062700  000002         12$:    ADD     #2,R0           ;Increment word index
010352  020027  000000C                CMP     R0,#PVNPW*2     ;Done all?
010356  103761                         BLO     10$             ;Loop if more
010360  005037  000000G                CLR     AFCF            ;Clear command file attribute flags
010364  032761  000000G 000000G        BIT     #$INKMN,LSW4(R1);Are we in TSKMON now?
010372  001572                         BEQ     9$              ;Br if not
010374  042761  000000G 000000G        BIC     #$SCCA,LSW5(R1) ;Clear abort-suppression flag
010402  042761  000000G 000000G        BIC     #$NOWIN,LSW11(R1);Clear window suppression
010410  105737  000000G                TSTB    SUCF2           ;Is there a pending secondary file?
010414  001561                         BEQ     9$              ;Br if not
010416  004737  000000G                CALL    STOP            ;Reenter KMON to start secondary file
                                ;
                                ;  Reopen next higher level file
```

```
                                        ;  Restore parameter pointers
                                        ;
010422  013705  000000G         2$:     MOV     CFSP,R5         ;GET COMMAND FILE STACK POINTER
010426  012703  000000G                 MOV     #PRMPNT,R3      ;POINT TO PARAM POINTER CELLS
010432  012504                  4$:     MOV     (R5)+,R4        ;GET A PARAMETER POINTER
010434  001402                          BEQ     3$              ;BR IF END OF LIST HIT
010436  010423                          MOV     R4,(R3)+        ;RESTORE POINTER
010440  000774                          BR      4$              ;GO BACK AND DO NEXT ONE
010442  020327  000000G         3$:     CMP     R3,#LSTPRM      ;ZERO ALL OTHER PARAMETER POINTERS
010446  103002                          BHIS    5$
010450  005023                          CLR     (R3)+
010452  000773                          BR      3$
                                        ;
                                        ;  Restore parameter strings
                                        ;
010454  012504                  5$:     MOV     (R5)+,R4        ;GET ADDRESS OF END OF STRING
010456  010437  000000G                 MOV     R4,PBFEND
010462  020427  000000G         7$:     CMP     R4,#PRMBUF      ;RESTORED ALL OF STRING?
010466  101402                          BLOS    6$              ;BR IF YES
010470  012544                          MOV     (R5)+,-(R4)     ;POP STRING OFF OF STACK
010472  000773                          BR      7$
010474  012537  000000G         6$:     MOV     (R5)+,CURPRM    ;POP POINTER INTO STRING
                                        ;
                                        ;  Restore IND status flags
                                        ;
010500  012537  000000G                 MOV     (R5)+,CFIND     ;RESTORE IND STATUS FLAGS
                                        ;
                                        ;  Reset command file control flags
                                        ;
010504  012704  000000G                 MOV     #CFLFL4,R4      ;GET CONTROL FLAG MASK
010510  040461  000000G                 BIC     R4,LSW4(R1)     ;CLEAR THOSE FLAGS
010514  005104                          COM     R4              ;MASK ALL BUT THOSE FLAGS
010516  040415                          BIC     R4,(R5)
010520  052561  000000G                 BIS     (R5)+,LSW4(R1)  ;SET DESIRED FLAGS
                                        ;
                                        ;  Restore command file attribute flags
                                        ;
010524  012537  000000G                 MOV     (R5)+,AFCF      ;Restore command file attribute flags
010530  052761  000000G 000000G         BIS     #$SCCA,LSW5(R1) ;Assume ctrl-C abort suppression wanted
010536  032737  000000G 000000G         BIT     #AF$CCA,AFCF    ;Is abort suppression wanted?
010544  001003                          BNE     13$             ;Br if yes
010546  042761  000000G 000000G         BIC     #$SCCA,LSW5(R1) ;Clear abort-suppression flag
010554  032737  000000G 000000G 13$:    BIT     #AF$NPW,AFCF    ;Suppressing process windowing?
010562  001003                          BNE     14$             ;Br if yes
010564  042761  000000G 000000G         BIC     #$NOWIN,LSW11(R1);Clear window suspend flag
                                        ;
                                        ;  Restore command file privileges
                                        ;
010572  005000                  14$:    CLR     R0              ;Init vector index
010574  011560  000000G         8$:     MOV     (R5),PRIVFO(R0) ;Restore each privilege word
010600  012560  000000G                 MOV     (R5)+,PRIVCO(R0);Reset current privilege too
010604  062700  000002                  ADD     #2,R0           ;Increment index
010610  020027  000000C                 CMP     R0,#PVNPW*2     ;Done all?
010614  103767                          BLO     8$              ;Loop if more
                                        ;
                                        ;  Restore buffer pointer information
                                        ;
```

```
    115 010616  012500                          MOV     (R5)+,R0        ;POINTER INTO BUFFER
    116 010620  004737  010124'                 CALL    CFSTRT          ;Reset command file pointer
    117 010624  012537  000000G                 MOV     (R5)+,CFBLK     ;CURRENT BLOCK NUMBER
    118                                 ;
    119                                 ;   Reopen the command file
    120                                 ;
    121 010630                                  .REOPEN #CFARG,#CFCHAN,R5 ;REOPEN COMMAND FILE
    122 010646  062705  000012                  ADD     #10.,R5         ;POP SAVE STATUS INFO
    123 010652  052761  000000G 000000G         BIS     #$CFOPN,LSW4(R1);SAY COMMAND FILE CHANNEL IS OPEN
    124                                 ;
    125                                 ;   Reread current buffer from file
    126                                 ;
    127 010660                                  .READW  #CFARG,#CFCHAN,#CFBUF,#256.,CFBLK
    128                                 ;
    129                                 ;   Finished restoring file
    130                                 ;
    131 010720  010537  000000G                 MOV     R5,CFSP         ;SAVE UPDATED STACK POINTER
    132 010724  105337  000000G                 DECB    CFNEST          ;SAY ONE LESS FILE ON STACK
    133                                 ;
    134                                 ;   If we just reached the end of a command file created to hold
    135                                 ;   an expanded IND command and KMON is trying to read another
    136                                 ;   command, call STOP to force KMON to go back to IND to get the
    137                                 ;   next command before continuing on with an outer level
    138                                 ;   command file.
    139                                 ;
    140 010730  032761  000000G 000000G         BIT     #$INKMN,LSW4(R1);IS KMON RUNNING?
    141 010736  001410                          BEQ     9$              ;BR IF NOT
    142 010740  013705  000000G                 MOV     CXTRMN,R5       ;GET ADDRESS OF SIMULATED RMON DATA
    143 010744  132765  000000G 000000G         BITB    #IN$ACT,R$INST(R5);IS IND ACTIVE?
    144 010752  001402                          BEQ     9$              ;BR IF NOT
    145 010754  004737  000000G                 CALL    STOP            ;REENTER KMON TO FORCE REENTRY OF IND
    146 010760  112637  000052          9$:     MOVB    (SP)+,@#52      ;RESTORE I/O ERROR CELL
    147 010764  012605                          MOV     (SP)+,R5
    148 010766  012604                          MOV     (SP)+,R4
    149 010770  012603                          MOV     (SP)+,R3
    150 010772  012601                          MOV     (SP)+,R1
    151 010774  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.
                                ;
010776  010046                  LOGCHR: MOV     R0,-(SP)
                                ;
                                ;  See if this is a control character
                                ;
011000  042700  177400                  BIC     #^C377,R0       ;Strip character down to 8 bits
011004  120027  000003                  CMPB    R0,#3           ;Is character control-C?
011010  001403                          BEQ     1$              ;Br if yes
011012  120027  000032                  CMPB    R0,#32          ;Is character control-Z?
011016  001007                          BNE     2$              ;Br if not
011020  112700  000136          1$:     MOVB    #'^,R0          ;Log "^"
011024  004737  011074'                 CALL    LOGCH1
011030  011600                          MOV     (SP),R0         ;Get back original character
011032  062700  000100                  ADD     #100,R0         ;Convert to printing character
011036  004737  011074'          2$:     CALL    LOGCH1          ;Log the character
                                ;
                                ;  Finished
                                ;
011042  012600                          MOV     (SP)+,R0
011044  000207                          RETURN
                                ;-----------------------------------------------------------------
                                ;  LOGCR is called to send a carriage-return and line-feed sequence
                                ;  to the log file.
                                ;
011046  010046                  LOGCR:  MOV     R0,-(SP)
011050  112700  0000000                 MOVB    #CR,R0          ;Log carriage-return
011054  004737  011074'                 CALL    LOGCH1
011060  112700  0000000                 MOVB    #LF,R0          ;Log line-feed
011064  004737  011074'                 CALL    LOGCH1
011070  012600                          MOV     (SP)+,R0
011072  000207                          RETURN
```

```
                                ;-----------------------------------------------------------------------
                                ;  LOGCH1 is called to move a character to the log buffer.
                                ;  No tests or conversions are performed on the character.
                                ;  No logging is done if echo suppression is in effect.
                                ;
                                ;  Inputs:
                                ;   R0 = Character to be written.
                                ;
011074  010246                  LOGCH1: MOV     R2,-(SP)
                                ;
                                ;  Check to see if log file output has been suspended (NOWRITE option)
                                ;
011076  032737  000000G 000000G         BIT     #LF$WRT,LOGFLG  ;Has log file been suspended?
011104  001453                          BEQ     9$              ;Br if yes
                                ;
                                ;  Get current buffer pointer and make sure buffer is not full
                                ;
011106  013702  000000G                 MOV     LOGPTR,R2       ;Get current log file buffer pointer
011112  001450                          BEQ     9$              ;Br if not doing logging
011114  020227  000000G                 CMP     R2,#LOGEND      ;Is buffer full?
011120  103442                          BLO     1$              ;Br if not
                                ;
                                ;  Log file buffer is full.  Write it to the log file.
                                ;
011122  012702  000000G                 MOV     #LOGBUF,R2      ;Point to front of buffer
011126  010046                          MOV     R0,-(SP)
011130  113746  000052                  MOVB    @#52,-(SP)      ;Save job's error cell
011134                                  .WRITW  #CFARG,#LOGCHN,R2,#256.,LOGBLK ;Write block to log file
011172  103010                          BCC     2$              ;Br if no write error
011174  012705  177753                  MOV     #-25,R5         ;Abort job if error writing to log file
011200  013704  000000G                 MOV     EMTADR,R4       ;Get address of EMT
011204  005037  000000G                 CLR     LOGPTR          ;Say we have stopped using log file
011210  000137  000000G                 JMP     ABORT           ;Abort the job
011214  005237  000000G         2$:     INC     LOGBLK          ;Advance log file block number
011220  112637  000052                  MOVB    (SP)+,@#52      ;Restore error cell
011224  012600                          MOV     (SP)+,R0
                                ;
                                ;  Move character to log buffer
                                ;
011226  110022                  1$:     MOVB    R0,(R2)+        ;Move char to log file buffer
011230  010237  000000G                 MOV     R2,LOGPTR       ;Save new buffer pointer
                                ;
                                ;  Finished
                                ;
011234  012602                  9$:     MOV     (SP)+,R2
011236  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 #.
                                ;
011240  032761  000000G 000000G ILWAIT: BIT     #$DETCH,LSW(R1) ;IS THIS A DETACHED JOB?
011246  001405                          BEQ     15$             ;BR IF NOT
011250  052761  000000G 000000G         BIS     #$DISCN,LSW(R1) ;FORCE JOB LOGOFF
011256  004737  000000G                 CALL    STOP            ;HALT DETACHED JOBS THAT WANT TERMINAL INPUT
                                ;
                                ;  If we previously stopped input from silo buffer, Restart it now
                                ;  if we are about to run out of characters.
                                ;
011262  032761  000000G 000000G 15$:    BIT     #$XSTOP,LSW6(R1);DID WE SUSPEND TRANSMISSION TO US?
011270  001410                          BEQ     11$             ;BR IF NOT
011272  042761  000000G 000000G         BIC     #$XSTOP,LSW6(R1);RESTART INPUT
011300  052761  000000G 000000G         BIS     #$NDICP,LSW10(R1);Say line needs input character servicing
011306  005237  000000G                 INC     NEDCDI          ;Say input processing needed
                                ;
                                ;  If we are in deferred echo mode, echo any pending
                                ;  Characters on last line.
                                ;
011312  004737  011430'         11$:    CALL    DFRREL          ;Release deferred echo mode
                                ;
                                ;  Suspend user till activation character received.
                                ;
011316  004737  017574'                 CALL    SIGWAT          ;SIGNAL VIRTUAL LINE WAIT CONDITION
011322  042761  000000G 000000G         BIC     #$NOIN,LSW3(R1) ;ALLOW INPUT TO BE ACCEPTED FOR LINE
011330  042761  000000G 000000G         BIC     #$SUCF,LSW9(R1) ;Say we are no longer executing startup file
011336  004737  000000G         13$:    CALL    CHKABT          ;SEE IF JOB HAS BEEN ABORTED
011342                                  DISABL                  ;** DISABLE **
011350  005761  000000G                 TST     LACTIV(R1)      ;GOT ANY ACTIVATION CHARS YET?
011354  001021                          BNE     1$              ;BR IF YES
011356  032761  000000G 000000G         BIT     #$NOINT,LSW7(R1);Does user want non-interactive execution?
011364  001006                          BNE     5$              ;Br if yes
011366  013761  000000G 000000G         MOV     VINTIO,LHIPCT(R1);RESET INTERACTIVE I/O LIMIT FOR JOB
011374  013761  000000G 000000G         MOV     VQUAN1,LITIME(R1);RESET INTERACTIVE CPU TIME LIMIT
011402  010046                  5$:     MOV     R0,-(SP)
011404  012700  000000G                 MOV     #S$INWT,R0      ;GET WAITING-FOR-INPUT STATE
011410  004737  000000G                 CALL    QHDSPN          ;SUSPEND JOB AND WAIT FOR ACTIVATION *ENABLE*
011414  012600                          MOV     (SP)+,R0
011416  000747                          BR      13$             ;NOW CHECK AGAIN
011420                          1$:     ENABL                   ;** ENABLE **
011426  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
                                ;
011430  010246                  DFRREL: MOV     R2,-(SP)
011432  010346                          MOV     R3,-(SP)
011434  010446                          MOV     R4,-(SP)
                                ;
                                ;  See if we are currently doing deferred echoing
                                ;
011436                                  DISABL                  ;** DISABLE INTERRUPTS **
011444  032761  000000G 000000G        BIT     #$DODFR,LSW3(R1);ARE WE IN DEFERRED ECHO MODE?
011452  001460                        BEQ     6$              ;BRANCH IF NOT
011454  005761  000000G               TST     LACTIV(R1)      ;ARE THERE ANY PENDING ACTIVATION CHARS?
011460  001055                        BNE     6$              ;BR IF YES -- DON'T ECHO IF PENDING ACTIV
011462                                  ENABL                   ;** ENABLE INTERRUPTS **
                                ;
                                ;  We are in deferred echo mode
                                ;
011470  042761  000000G 000000G        BIC     #$GCECO,LSW3(R1);RESET GETCHR ECHOING
011476  052761  000000G 000000G        BIS     #$1STCH,LSW3(R1);SAY WE'VE GOT 1ST CHAR
011504  116161  000000G 000000G        MOVB    LCOL(R1),LINCUR(R1);SAVE CURSOR POSITION
011512  005004                        CLR     R4              ;R4 IS FLAG FOR ESC SEQ CHARS
011514  005003                        CLR     R3
011516  016102  000000G               MOV     LINPNT(R1),R2   ;POINT TO START OF PENDING CHARS
                                ;
                                ;  Begin loop to echo all pending characters
                                ;
011522  020361  000000G         5$:    CMP     R3,LINCNT(R1)   ;ECHOED ALL PENDING?
011526  103027                        BHIS    4$              ;BRANCH IF FINISHED
011530  004737  016420'               CALL    FETCHR          ;GET NEXT CHAR FROM TT INPUT BUFFER
011534  005704                        TST     R4              ;IS THIS CHAR PART OF ESC SEQ?
011536  001015                        BNE     8$              ;BR IF YES
011540  120027  000000G               CMPB    R0,#ESCFLG      ;FLAG THAT NEXT CHAR PART OF ESC SEQUENCE?
011544  001006                        BNE     9$              ;BR IF NOT
011546  032761  000000G 000000G        BIT     #$VTESC,LSW5(R1);Are we activating on escape sequences?
011554  001402                        BEQ     9$              ;Br if not -- Treat ESCFLG like normal char
011556  005204                        INC     R4              ;SET ESC SEQ FLAG
011560  000410                        BR      10$
011562  004737  017522'          9$:    CALL    CVTLC           ;CONVERT LOWER CASE TO UPPER CASE
011566  004737  007100'               CALL    GCECHO          ;ECHO IT
011572  005004                  8$:    CLR     R4              ;CLEAR ESC SEQ FLAG
011574  032700  000000G               BIT     #ACFLAG,R0      ;IS THIS AN ACTIVATION CHAR?
011600  001005                        BNE     6$              ;BRANCH IF YES
011602  005203                  10$:   INC     R3              ;COUNT CHARS ECHOED
011604  000746                        BR      5$
                                ;
                                ;  Release deferred echo mode
                                ;
011606  042761  000000G 000000G 4$:    BIC     #$DODFR,LSW3(R1);BEGIN IMMEDIATE CHAR ECHOING
                                ;
                                ;  Finished
                                ;
```

```
     58 011614                          6$:     ENABL                   ;** ENABLE INTERRUPTS **
     59 011622  012604                          MOV     (SP)+,R4
     60 011624  012603                          MOV     (SP)+,R3
     61 011626  012602                          MOV     (SP)+,R2
     62 011630  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
                                ;
011632  010146                  TTINCP: MOV     R1,-(SP)
011634  010546                          MOV     R5,-(SP)
011636  010401                          MOV     R4,R1           ;Carry line index number in R1 too
                                ;
                                ;  See if TT input ring buffer is full
                                ;
011640  032764  000000G 000000G 5$:     BIT     #$XSTOP,LSW6(R4);Has input been suspended due to buffer full
011646  001007                          BNE     9$              ;Br if yes
                                ;
                                ;  Get next character from input silo
                                ;
011650  004777  000000G                 CALL    @SILFET         ;Get next character from input silo
011654  103404                          BCS     9$              ;Br if no characters in silo
011656  010005                          MOV     R0,R5           ;Put character in R5 for GOTCHR
                                ;
                                ;  Now call main routine to process a character for this line
                                ;
011660  004737  011674'                 CALL    GOTCHR          ;Process a character for this line
                                ;
                                ;  Finished processing a character.
                                ;  Go back and see if there are any more characters to process.
                                ;
011664  000765                          BR      5$              ;Check for more characters to process
                                ;
                                ;  Finished processing all pending characters for this line.
                                ;
                                ;
011666  012605                  9$:     MOV     (SP)+,R5
011670  012601                          MOV     (SP)+,R1
011672  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.
                                ;
011674  010146                  GOTCHR: MOV     R1,-(SP)
                                ;
                                ;  Ignore the character if system initialization is not yet complete
                                ;
011676  105737  000000G                 TSTB    INITFLG         ;Is system initialization finished yet?
011702  001052                          BNE     9$              ;Br if not
                                ;
                                ;  See if this line has been initialized yet
                                ;
011704  116401  000000G                 MOVB    LNMAP(R4),R1    ;Get virtual line index number
011710  032761  000000G 000000G         BIT     #$DILUP,LSW(R1) ;Has line been started yet?
011716  001033                          BNE     1$              ;Br if yes
                                ;
                                ;  Line has not been initialized yet.
                                ;  See if we should start it now.
                                ;
011720  105737  000000G                 TSTB    STPFLG          ;Is a system shutdown in progress?
011724  001041                          BNE     9$              ;Br if yes -- Don't start any lines
011726  032764  000000G 000000G         BIT     #$DEAD,LSW3(R4) ;Is this line marked as dead?
011734  001035                          BNE     9$              ;Br if yes
                                ;
                                ;  See if we should do autobaud speed selection for this line
                                ;
011736  032764  000000G 000000G         BIT     #$AUTO,ILSW2(R4);Is autobaud speed selection wanted?
011744  001405                          BEQ     3$              ;Br if not
011746                                  OCALL   AUTSPD          ;Do autobaud speed selection
011754  103007                          BCC     2$              ;Br if we should start the line now
011756  000424                          BR      9$              ;Do not start the line yet
011760  120527  000000G         3$:     CMPB    R5,#CR          ;Is character Carriage-return?
011764  001403                          BEQ     2$              ;Br if yes
011766  120527  000000G                 CMPB    R5,#CTRLC       ;Is character ctrl-C?
011772  001016                          BNE     9$              ;Br if not
                                ;
                                ;  Start up a previously inactive line
                                ;
011774  005000                  2$:     CLR     R0              ;No secondary start-up command file
011776                                  OCALL   INITLN          ;Initialize the line
012004  000403                          BR      5$
                                ;
                                ;  See if we are ignoring trash characters that may be received after
                                ;  the autobaud start-up character.
                                ;
012006  005764  000000G         1$:     TST     LABTIM(R4)      ;Is autobaud masking trash characters?
012012  001404                          BEQ     4$              ;Br if not
012014  042761  000000G 000000G 5$:     BIC     #$RBRK,LSW10(R1);Clear break-received flag
012022  000402                          BR      9$              ;Wait for autobaud mask time to end
                                ;
                                ;  Process an input character for an active line
                                ;
```

```
     58 012024  004737  012034'          4$:     CALL    PRCHAR          ;Process the character
     59                                   ;
     60                                   ;  Finished
     61                                   ;
     62 012030  012601                    9$:     MOV     (SP)+,R1
     63 012032  000207                            RETURN
```

```
                                ;-----------------------------------------------------------------------
                                ;  Process a character received by an active line.
                                ;
                                ;  Inputs:
                                ;   R1 = Job's virtual line index number.
                                ;   R5 = Received character.
                                ;
012034  010246                  PRCHAR: MOV     R2,-(SP)
012036  010346                          MOV     R3,-(SP)
012040  010446                          MOV     R4,-(SP)
                                ;
                                ;  See if we received a real break (long space).
                                ;
012042  016104  000000G                 MOV     LNPRIM(R1),R4   ;Get primary line #
012046  032764  000000G 000000G         BIT     #$RBRK,LSW10(R4);Did we receive a break?
012054  001406                          BEQ     1$              ;Br if not
012056  004737  017652'                 CALL    SIGBRK          ;Signal that we received a break
012062  042764  000000G 000000G         BIC     #$RBRK,LSW10(R4);Clear break-received flag
012070  000554                          BR      PRCEND          ;Finished with break character
                                ;
                                ;  Mask character to 8 bits and ignore nulls
                                ;
012072  042705  177400          1$:     BIC     #^C377,R5       ;Mask character to 8 bits
012076  001551                          BEQ     PRCEND          ;Ignore null characters
                                ;
                                ;  If debugger is in control, bypass some special character processing
                                ;
012100  032761  000000G 000000G         BIT     #$DBGMD,LSW6(R1);Is debugging program in control?
012106  001054                          BNE     CKCW            ;If yes then bypass some checking
                                ;
                                ;  See if user defined an asynchronous break character
                                ;
012110  016102  000000G                 MOV     LBRKCH(R1),R2   ;Did user define an asynch break char?
012114  001405                          BEQ     CKSPAC          ;Br if not
012116  020502                          CMP     R5,R2           ;Is this the break character?
012120  001003                          BNE     CKSPAC          ;Br if not
012122  004737  017652'                 CALL    SIGBRK          ;Tell user that break char was received
012126  000535                          BR      PRCEND
                                ;
                                ;  See if this is a user-defined activation character
                                ;
012130  016102  000000G         CKSPAC: MOV     LNSPAC(R1),R2   ;Get number of user-defined activation chars
012134  001424                          BEQ     CKVTES          ;Br if there are none
012136  010500                          MOV     R5,R0           ;Get current character
012140  004737  017522'                 CALL    CVTLC           ;Convert to upper-case if needed
012144  016103  000000G                 MOV     LSPACT(R1),R3   ;Get pointer to table of activation chars
012150  120023                  1$:     CMPB    R0,(R3)+        ;Is this an activation character?
012152  001402                          BEQ     2$              ;Br if yes
012154  077203                          SOB     R2,1$           ;Loop if more to check
012156  000404                          BR      CKHIIN          ;This is not a user-defined activation char
012160  010005                  2$:     MOV     R0,R5           ;Get converted character to R5
012162  004737  016300'                 CALL    STRACT          ;Store the activation character
012166  000515                          BR      PRCEND
                                ;
                                ;  See if we are in high-efficienty mode
                                ;
012170  032761  000000G 000000G CKHIIN: BIT     #$HITTY,LSW4(R1);Are we in high-efficiency mode?
```

```
     58 012176  001403                              BEQ     CKVTES          ;Br if not
     59 012200  004737  016002'                     CALL    STRCHR          ;Store the character
     60 012204  000506                              BR      PRCEND
     61                                 ;
     62                                 ;  See if this is a VT52 escape-letter sequence
     63                                 ;
     64 012206  032761  000000G 000000G CKVTES:     BIT     #$VTESC,LSW5(R1);Activate on esc-letter sequence?
     65 012214  001411                              BEQ     CKCW            ;Br if not
     66 012216  004737  017322'                     CALL    SCACHK          ;Are we in single character activ mode?
     67 012222  103406                              BCS     CKCW            ;Br if yes
     68 012224  005761  000000G                     TST     LTTCR(R1)       ;Is terminal input cmpl rtn scheduled?
     69 012230  001003                              BNE     CKCW            ;Br if yes
     70 012232  004737  015310'                     CALL    CKVTAC          ;Check for escape-letter sequence
     71 012236  103071                              BCC     PRCEND          ;Br if char was part of escape sequence
     72                                 ;
     73                                 ;  Check for request to switch to a virtual line
     74                                 ;
     75 012240  120537  000000G         CKCW:       CMPB    R5,VVLSCH       ;Is this char a request to switch to vir line?
     76 012244  001024                              BNE     1$              ;Br if not
     77 012246  032761  000000G 000000G            BIT     #$CTRLW,LSW3(R1);Was last character also control-W?
     78 012254  001414                              BEQ     2$              ;Br if not
     79 012256  042761  000000G 000000G            BIC     #$1ESC,LSW(R1)  ;Say last char was not escape
     80 012264  042761  000000G 000000G            BIC     #$CTRLW,LSW3(R1);Say last char not control-W
     81 012272  004737  017504'                     CALL    SCACHR          ;See if we are in single-char activation mode
     82 012276  103451                              BCS     PRCEND          ;Br if in single-char activation mode
     83 012300  004737  015764'                     CALL    INCHR           ;Pass control-W to program as normal char
     84 012304  000446                              BR      PRCEND
     85 012306  052761  000000G 000000G 2$:        BIS     #$CTRLW,LSW3(R1);Remember last char was control-W
     86 012314  000442                              BR      PRCEND
     87 012316  032761  000000G 000000G 1$:        BIT     #$CTRLW,LSW3(R1);Was control-W the last character?
     88 012324  001420                              BEQ     CKICTL          ;Br if not
     89 012326  020527  000037                      CMP     R5,#37          ;Is current character a control character?
     90 012332  101415                              BLOS    CKICTL          ;Br if yes
     91 012334  042761  000000G 000000G            BIC     #$CTRLW,LSW3(R1);Say control-W is not the last char
     92 012342  162705  000060                      SUB     #'0,R5          ;Convert line # digit to binary value
     93 012346  002425                              BLT     PRCEND          ;Br if too small
     94 012350  020527  000000G                     CMP     R5,#MAXSEC      ;Don't exceed max line # allowed
     95 012354  003022                              BGT     PRCEND          ;Br if too large
     96 012356                                      OCALL   DOSWIT          ;Switch to a virtual line
     97 012364  000416                              BR      PRCEND
     98                                 ;
     99                                 ;  Determine if this is a normal or control character
    100                                 ;
    101 012366  020527  000037         CKICTL:     CMP     R5,#37          ;Is this a normal or control char?
    102 012372  101003                              BHI     1$              ;Br if not control character
    103 012374  004737  012650'                     CALL    DOCTRL          ;Process a control character
    104 012400  000410                              BR      PRCEND
    105 012402  120527  000000G         1$:         CMPB    R5,#RUBOUT      ;Is this a rubout character?
    106 012406  001003                              BNE     2$              ;Br if not
    107 012410  004737  014752'                     CALL    ICPRUB          ;Process rubout character
    108 012414  000402                              BR      PRCEND
    109 012416  004737  012432'         2$:         CALL    REGCHR          ;Process a normal character
    110                                 ;
    111                                 ;  Finished
    112                                 ;
    113 012422  012604                 PRCEND:     MOV     (SP)+,R4
    114 012424  012603                             MOV     (SP)+,R3
```

```
    115 012426  012602                          MOV     (SP)+,R2
    116 012430  000207                          RETURN
```

```
                                              .SBTTL  REGCHR -- Process normal characters
                                      ;-----------------------------------------------------------------------
                                      ;  Process normal (non-control) characters.
                                      ;
                                      ;  Inputs:
                                      ;   R1 = Virtual line index number.
                                      ;   R5 = Current input character.
                                      ;
 012432  010246                       REGCHR: MOV     R2,-(SP)
 012434  010546                               MOV     R5,-(SP)
 012436  010500                               MOV     R5,R0           ;Copy the character
                                      ;
                                      ;  Ignore all input while processing logoff command file
                                      ;
 012440  032761  000000G 000000G              BIT     #$LOFCF,LSW9(R1);Are we processing a logoff command file?
 012446  001075                               BNE     9$              ;Br if yes -- ignore the character
                                      ;
                                      ;  Say last character was not control-C
                                      ;
 012450  042761  000000G 000000G              BIC     #$1ESC,LSW(R1)  ;Say last char was not escape
 012456  042761  000000G 000000G              BIC     #$1CTLC,LSW5(R1);Say last character was not control-C
                                      ;
                                      ;  See if we are in single character activation mode
                                      ;
 012464  004737  017362'                      CALL    SLCHK           ;Are we in single-char activation mode or SL?
 012470  103003                               BCC     6$              ;Br if not
 012472  004737  016360'                      CALL    STRSNG          ;Store character and activate
 012476  000461                               BR      9$
                                      ;
                                      ;  See if ODT is in control
                                      ;
 012500  032761  000000G 000000G 6$:          BIT     #$ODTMD,LSW4(R1);Is ODT activation in effect?
 012506  001413                               BEQ     3$              ;Br if not
 012510  004737  015522'                      CALL    CHKODT          ;Is this an ODT activation char?
 012514  103010                               BCC     3$              ;Br if not
 012516  120027  000040                       CMPB    R0,#40          ;Is this a control character?
 012522  103402                               BLO     4$              ;Br if yes -- Don't echo it
 012524  004737  017140'                      CALL    ECHO            ;Echo the character
 012530  004737  016360'         4$:          CALL    STRSNG          ;Store character and activate
 012534  000442                               BR      9$
                                      ;
                                      ;  Check for activation on input of a certain number of characters
                                      ;
 012536  016102  000000G         3$:          MOV     LAFSIZ(R1),R2   ;Was field width activation specified?
 012542  001416                               BEQ     1$              ;Br if not
 012544  005302                               DEC     R2              ;Is field full yet?
 012546  020261  000000G                      CMP     R2,LINCNT(R1)
 012552  101012                               BHI     1$              ;Br if not
 012554  032761  000000G 000000G              BIT     #$DBGMD,LSW6(R1);Is a debugger running?
 012562  001006                               BNE     1$              ;Br if yes
 012564  004737  017140'                      CALL    ECHO            ;Echo character
 012570  010005                               MOV     R0,R5           ;Save the converted character
 012572  004737  016300'                      CALL    STRACT          ;Store the activation character
 012576  000421                               BR      9$
                                      ;
                                      ;  See if we need to limit number of characters that can be
                                      ;  typed into a field
```

```
     58                                     ;
     59 012600  016102  000000G             1$:     MOV     LFWLIM(R1),R2   ;If field width limit specified?
     60 012604  001414                              BEQ     2$              ;Br if not
     61 012606  026102  000000G                     CMP     LINCNT(R1),R2   ;Would this character overflow the field?
     62 012612  103411                              BLO     2$              ;Br if not
     63 012614  032761  000000G 000000G             BIT     #$DBGMD,LSW6(R1);Is debugger in control?
     64 012622  001005                              BNE     2$              ;Br if yes
     65 012624  012700  000000G                     MOV     #BELL,R0        ;Echo bell
     66 012630  004737  017200'                     CALL    ECHO2
     67 012634  000402                              BR      9$              ;Discard the character
     68                                     ;
     69                                     ;  Normal character being input
     70                                     ;
     71 012636  004737  015764'             2$:     CALL    INCHR           ;Store and echo the character
     72                                     ;
     73                                     ;  Finished
     74                                     ;
     75 012642  012605                      9$:     MOV     (SP)+,R5
     76 012644  012602                              MOV     (SP)+,R2
     77 012646  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.
                                ;
012650                          DOCTRL:
                                ;
                                ;  See if this is a request for job status information
                                ;
012650  120537  0000006                 CMPB    R5,VCTRLT       ;Request for ^T info?
012654  001004                          BNE     3$              ;Br if not
012656                                  OCALL   DOCTLT          ;Call TSTTY2 to do it
012664  000425                          BR      9$              ;
                                ;
                                ;  See if this is a request to print the current window
                                ;
012666  120537  0000006        3$:      CMPB    R5,VVPWCH       ;Request to print screen?
012672  001016                          BNE     1$              ;Br if not
012674  032761  0000006 0000006         BIT     #$PWKEY,LSW11(R1);Is print window control char enabled?
012702  001412                          BEQ     1$              ;Br if not
012704  016100  0000006               MOV     LWINDO(R1),R0   ;Is windowing enabled for this process?
012710  001407                          BEQ     1$              ;Br if not -- Treat char like ordinary char
                                ;
                                ;  This is a request to print the current window contents
                                ;
012712  010246                          MOV     R2,-(SP)
012714  010002                          MOV     R0,R2           ;Get address of current window control blk
012716                                  OCALL   WINPRT          ;Print the window
012724  012602                 2$:      MOV     (SP)+,R2
012726  000404                          BR      9$
                                ;
                                ;  This is an ordinary control character
                                ;
012730  010500                 1$:      MOV     R5,R0           ;Get the control character
012732  006300                          ASL     R0              ;Convert to word table index
012734  004770  012742'                 CALL    @CTLRTN(R0)     ;Call appropriate processing routine
                                ;
                                ;  Finished
                                ;
012740  000207                 9$:      RETURN
                                ;
                                ;-----------------------------------------------------------------------
                                ;  Branch table for control character processing routines.
                                ;
012742  012432'                 CTLRTN:  .WORD   REGCHR          ; 00 - NUL
012744  012432'                          .WORD   REGCHR          ; 01 - SOH (control-A)
012746  012432'                          .WORD   REGCHR          ; 02 - STX (control-B)
012750  013302'                          .WORD   ICPCTC          ; 03 - ETX (control-C)
012752  013632'                          .WORD   ICPCTD          ; 04 - EOT (control-D)
012754  012432'                          .WORD   REGCHR          ; 05 - ENQ (control-E)
012756  012432'                          .WORD   REGCHR          ; 06 - ACK (control-F)
012760  013702'                          .WORD   ICPCTG          ; 07 - BEL (control-G)
012762  012432'                          .WORD   REGCHR          ; 10 - Backspace
```

```
     58 012764  012432'                    .WORD   REGCHR          ; 11 - TAB (control-I)
     59 012766  013176'                    .WORD   ICPLF           ; 12 - Line feed
     60 012770  012432'                    .WORD   REGCHR          ; 13 - VT  (control-K)
     61 012772  012432'                    .WORD   REGCHR          ; 14 - FF  (control-L)
     62 012774  013042'                    .WORD   ICPCR           ; 15 - Carriage return
     63 012776  012432'                    .WORD   REGCHR          ; 16 - SO  (control-N)
     64 013000  013766'                    .WORD   ICPCTO          ; 17 - SI  (control-O)
     65 013002  012432'                    .WORD   REGCHR          ; 20 - DLE (control-P)
     66 013004  012432'                    .WORD   REGCHR          ; 21 - DC1 (control-Q)
     67 013006  014056'                    .WORD   ICPCTR          ; 22 - DC2 (control-R)
     68 013010  012432'                    .WORD   REGCHR          ; 23 - DC3 (control-S)
     69 013012  012432'                    .WORD   REGCHR          ; 24 - DC4 (control-T)
     70 013014  014122'                    .WORD   ICPCTU          ; 25 - NAK (control-U)
     71 013016  012432'                    .WORD   REGCHR          ; 26 - SYN (control-V)
     72 013020  012432'                    .WORD   REGCHR          ; 27 - ETB (control-W)
     73 013022  014502'                    .WORD   ICPCTX          ; 30 - CAN (control-X)
     74 013024  012432'                    .WORD   REGCHR          ; 31 - EM  (control-Y)
     75 013026  014566'                    .WORD   ICPCTZ          ; 32 - SUB (control-Z)
     76 013030  014622'                    .WORD   ICPESC          ; 33 - ESC
     77 013032  012432'                    .WORD   REGCHR          ; 34 - FS
     78 013034  012432'                    .WORD   REGCHR          ; 35 - GS
     79 013036  012432'                    .WORD   REGCHR          ; 36 - RS
     80 013040  012432'                    .WORD   REGCHR          ; 37 - US
```

```
                                        .SBTTL  ICPCR  -- Carriage-return processing
                                ;-----------------------------------------------------------------------
                                ;  Process Carriage-return character.
                                ;
                                ;  Inputs:
                                ;   R1 = Virtual line index number.
                                ;   R5 = Current input character.
                                ;
013042  010546                  ICPCR:  MOV     R5,-(SP)
                                ;
                                ;  See if we are in single-character activation mode
                                ;
013044  004737  017362'                 CALL    SLCHK           ;Are we in single-char activation mode?
013050  103005                          BCC     1$              ;Br if not
013052  004737  016300'                 CALL    STRACT          ;Pass carriage return to program
013056  112705  000000G                 MOVB    #LF,R5          ;and follow with line feed
013062  000433                          BR      3$
                                ;
                                ;  We are not in single-character activation mode.
                                ;  See if ODT is running.
                                ;
013064  032761  000000G 000000G 1$:     BIT     #$ODTMD,LSW4(R1);Is ODT in control?
013072  001410                          BEQ     4$              ;Br if not
013074  010500                          MOV     R5,R0           ;Get the CR character
013076  004737  017140'                 CALL    ECHO            ;Echo CR
013102  012700  000000G                 MOV     #LF,R0          ;Get LF character
013106  004737  017144'                 CALL    ECHO1           ;Echo LF
013112  000417                          BR      3$              ;Store CR as activation character
                                ;
                                ;  We are not in ODT mode
                                ;
013114  004737  015764'         4$:     CALL    INCHR           ;Store and echo CR
013120  112705  000000G                 MOVB    #LF,R5          ;We will follow CR with LF
013124  032761  000000G 000000G         BIT     #$DBGMD,LSW6(R1);Is a debug program running?
013132  001004                          BNE     2$              ;Br if yes
013134  032761  000000G 000000G         BIT     #$NOLF,LSW6(R1) ;Are we suppressing LF echoing after CR?
013142  001003                          BNE     3$              ;Br if yes
013144  010500                  2$:     MOV     R5,R0           ;Get character to echo
013146  004737  017144'                 CALL    ECHO1           ;Echo line-feed
013152  004737  016300'         3$:     CALL    STRACT          ;Store activation character
                                ;
                                ;  Finished
                                ;
013156  042761  000000G 000000G 9$:     BIC     #$1CTLC,LSW5(R1);Say last character was not control-C
013164  042761  000000G 000000G         BIC     #$1ESC,LSW(R1)  ;Say last char was not escape
013172  012605                          MOV     (SP)+,R5
013174  000207                          RETURN
```

```
                                        .SBTTL  ICPLF  -- Line-feed processing
                                ;-----------------------------------------------------------------------
                                ;  Process Line-feed input character.
                                ;
                                ;  Inputs:
                                ;   R1 = Virtual line index number.
                                ;   R5 = Current input character.
                                ;
013176  010546                  ICPLF:  MOV     R5,-(SP)
                                ;
                                ;  If we are in "paper-tape" mode, ignore line-feed characters.
                                ;
013200  032761  000000G 000000G         BIT     #$TAPE,LSW2(R1) ;Are we in paper-tape mode?
013206  001025                          BNE     9$              ;Br if yes
                                ;
                                ;  See if we are in single-character activation mode
                                ;
013210  004737  017504'                 CALL    SCACHR          ;See if we are in single-char activation mode
013214  103422                          BCS     9$              ;Br if yes
                                ;
                                ;  We are not in single-character activation mode.
                                ;  See if we are in ODT mode.
                                ;
013216  032761  000000G 000000G         BIT     #$ODTMD,LSW4(R1);Is ODT in control?
013224  001412                          BEQ     2$              ;Br if not
013226  112700  000000G                 MOVB    #CR,R0          ;Echo carriage return
013232  004737  017140'                 CALL    ECHO
013236  110500                          MOVB    R5,R0           ;Get line feed character
013240  004737  017144'                 CALL    ECHO1           ;Echo line-feed
013244  004737  016300'                 CALL    STRACT          ;Store line feed and activate
013250  000404                          BR      9$
                                ;
                                ;  Treat line-feed exactly like carriage-return.
                                ;
013252  112705  000000G         2$:     MOVB    #CR,R5          ;Get CR character
013256  004737  013042'                 CALL    ICPCR           ;Process the carriage-return
                                ;
                                ;  Finished
                                ;
013262  042761  000000G 000000G 9$:     BIC     #$1CTLC,LSW5(R1);Say last character was not control-C
013270  042761  000000G 000000G         BIC     #$1ESC,LSW(R1)  ;Say last char was not escape
013276  012605                          MOV     (SP)+,R5
013300  000207                          RETURN
```

```
                                        .SBTTL  ICPCTC -- Control-C processing
                                ;-----------------------------------------------------------------------
                                ;  Process a control-C input character.
                                ;
                                ;  Inputs:
                                ;   R1 = Virtual line index number.
                                ;   R5 = Current input character.
                                ;
013302  010246                  ICPCTC: MOV     R2,-(SP)
013304  010446                          MOV     R4,-(SP)
                                ;
                                ;  Determine if we have received 2 consecutive control-C characters.
                                ;
013306  032761  000000G 000000G         BIT     #$1CTLC,LSW5(R1);Was last character control-C?
013314  001021                          BNE     2$              ;Br if yes
                                ;
                                ;  This is 1st control-C
                                ;
013316  052761  000000G 000000G         BIS     #$1CTLC,LSW5(R1);Remember last char was control-C
013324  042761  000000G 000000G 7$:     BIC     #$1ESC,LSW(R1)  ;Say last char was not escape
013332  042761  000000G 000000G         BIC     #$RTCS,LSW9(R1) ;Say we are not receiving control sequence
013340  004737  017504'                 CALL    SCACHR          ;See if we are in single-char activation mode
013344  103527                          BCS     9$              ;Br if yes
013346  004737  017216'                 CALL    ECOCTL          ;Echo "^C"
013352  004737  016300'                 CALL    STRACT          ;Store control-C as activation char
013356  000522                          BR      9$
                                ;
                                ;  We received two consecutive control-C characters
                                ;
013360  032761  000000C 000000G 2$:     BIT     #<C$SUCF!$LOFCF>,LSW9(R1);Are we doing logon/logoff processing?
013366  001356                          BNE     7$              ;Br if yes -- don't allow ctrl-C abort here
013370  032761  000000G 000000G         BIT     #$SCCA,LSW5(R1) ;Suppressing control-C aborts for program?
013376  001352                          BNE     7$              ;Br if yes
013400  032761  000000G 000000G         BIT     #$NOIN,LSW3(R1) ;Special no-input flag set?
013406  001346                          BNE     7$              ;Br if yes
013410  032761  000000G 000000G 21$:    BIT     #$RFRSH,LSW4(R1);Is a window refresh being done now?
013416  001342                          BNE     7$              ;Don't allow abort during refresh
013420  016102  000000G                 MOV     LSCCA(R1),R2    ;Did user do .SCCA EMT?
013424  001412                          BEQ     3$              ;Br if not
013426  032761  000000G 000000G         BIT     #$DBGMD,LSW6(R1);Is debugger in control?
013434  001006                          BNE     3$              ;Br if yes -- ignore .SCCA
                                ;
                                ;  User did a .SCCA -- Set flag to remember to tell him about ctrl-C's
                                ;
013436  052761  000000G 000000G         BIS     #$SETCC,LSW4(R1);Remember to tell him later
013444  105237  000000G                 INCB    DOSCHD          ;Request a job scheduler cycle
013450  000725                          BR      7$
                                ;
                                ;  User did not do a .SCCA so abort him
                                ;
013452  042761  000000C 000000G 3$:     BIC     #<LCBIT!SPCTTY>,LJSW(R1) ;Clean out JSW
013460  016161  000000G 000000G         MOV     LINNXT(R1),LINPNT(R1) ;Kill all input
013466  016161  000000G 000000G         MOV     LINNXT(R1),LSTACT(R1)
013474  005061  000000G                 CLR     LINCNT(R1)
013500  016161  000000G 000000G         MOV     LINSIZ(R1),LINSPC(R1)
013506  016161  000000G 000000G         MOV     LOTSIZ(R1),LOTSPC(R1) ;Kill all output
013514  016161  000000G 000000G         MOV     LOTNXT(R1),LOTPNT(R1)
```

```
      58 013522  005061  000000G                 CLR     LFWLIM(R1)          ;No field width limit
      59 013526  005061  000000G                 CLR     LAFSIZ(R1)          ;No field width activation
      60 013532  042761  000000G 000000G         BIC     #$SLINI,LSW7(R1);Say SL must reinitialize for next line
      61 013540  042761  000000C 000000G         BIC     #<$DODFR!$GCECO>,LSW3(R1) ;Reset deferred echoing
      62 013546  016104  000000G                 MOV     LNPRIM(R1),R4       ;Get primary job number
      63 013552  042764  000000G 000000G         BIC     #$CTRLS,LSW3(R4);Reset control-S output suspension
      64 013560  004737  017216'                 CALL    ECOCTL              ;Echo "^C"
      65 013564  052761  000000G 000000G         BIS     #$CTRLC,LSW(R1) ;Set job abort flag
      66 013572  052761  000000G 000000G         BIS     #$CFKIL,LSW6(R1);Set flag to abort all open command files
      67 013600  042761  000000G 000000G         BIC     #$SUSPN,LSW(R1) ;Clear job-suspended flag
      68 013606  004737  000000G                 CALL    FORCEX              ;Force execution of the job
      69 013612  005061  000000G                 CLR     LACTIV(R1)          ;Say no chars received yet
      70 013616  042761  000000G 000000G         BIC     #$1STCH,LSW3(R1)
      71                                 ;
      72                                 ;  Finished
      73                                 ;
      74 013624  012604                  9$:    MOV     (SP)+,R4
      75 013626  012602                         MOV     (SP)+,R2
      76 013630  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.
                                ;
013632  032761  000000G 000000G ICPCTD: BIT     #$INKMN,LSW4(R1);Are we running in TSKMON now?
013640  001405                         BEQ     2$              ;Br if not
013642  032761  000000G 000000G        BIT     #$DBKMN,LSW9(R1);Should we debug TSKMON?
013650  001411                         BEQ     1$              ;Br if not
013652  000404                         BR      3$              ;Yes, go force breakpoint
013654  032761  000000C 000000G 2$:    BIT     #$DEBUG!$CTRLD,LSW9(R1);Is job running with the debugger?
013662  001404                         BEQ     1$              ;Br if not
013664  052761  000000G 000000G 3$:    BIS     #$DBGBK,LSW9(R1);Set flag to force debug breakpoint
013672  000402                         BR      9$
                                ;
                                ;  We are not running with debugger, treat Ctrl-D like normal character
                                ;
013674  004737  012432'         1$:     CALL    REGCHR          ;Store as regular character
                                ;
                                ;  Finished
                                ;
013700  000207                 9$:     RETURN
```

```
                                        .SBTTL  ICPCTG -- Control-G processing
                                ;-----------------------------------------------------------------------
                                ;  Process a Control-G (Bell) character.
                                ;
                                ;  Inputs:
                                ;   R1 = Virtual line index number.
                                ;   R5 = Current input character.
                                ;
013702  032761  000000G 000000G ICPCTG: BIT     #SPCTTY,LJSW(R1);Is special character processing wanted?
013710  001415                          BEQ     2$              ;Br if not
013712  032761  000000G 000000G        BIT     #$CHACT,LSW5(R1);Is single character activation wanted?
013720  001403                          BEQ     1$              ;Br if not
013722  004737  016360'                CALL    STRSNG          ;Store the character
013726  000410                          BR      9$
                                ;
                                ;  Special-TTY mode is set but not single character activation
                                ;
013730  010500                  1$:     MOV     R5,R0           ;Get char to echo
013732  004737  017140'                CALL    ECHO            ;Echo a bell
013736  004737  016300'                CALL    STRACT          ;Store and activate
013742  000402                          BR      9$
                                ;
                                ;  Normal character processing
                                ;
013744  004737  012432'          2$:     CALL    REGCHR          ;Treat as regular character
                                ;
                                ;  Finished
                                ;
013750  042761  000000G 000000G 9$:      BIC     #$1CTLC,LSW5(R1);Say last character was not control-C
013756  042761  000000G 000000G         BIC     #$1ESC,LSW(R1)  ;Say last char was not escape
013764  000207                          RETURN
```

```
                                        .SBTTL  ICPCTO -- Control-O processing
                                ;-----------------------------------------------------------------
                                ;  Process a Control-O input character.
                                ;
                                ;  Inputs:
                                ;   R1 = Virtual line index number.
                                ;   R5 = Current input character.
                                ;
013766  032761  000000G 000000G ICPCTO: BIT     #$CTRLO,LSW3(R1);Is output currently suppressed?
013774  001022                         BNE     1$              ;Br if yes (reenable it)
                                ;
                                ;  Begin suppressing output
                                ;
013776  016161  000000G 000000G        MOV     LOTSIZ(R1),LOTSPC(R1) ;Empty output buffer
014004  016161  000000G 000000G        MOV     LOTNXT(R1),LOTPNT(R1)
014012  004737  017216'               CALL    ECOCTL          ;Echo "^O"
014016  052761  000000G 000000G        BIS     #$CTRLO,LSW3(R1);Remember output is being suppressed
014024  026127  000000G 000000G        CMP     LSTATE(R1),#S$OTWT ;Is job waiting for output space?
014032  001010                         BNE     2$              ;Br if not
014034  004737  006000'               CALL    OTREQO          ;Reactivate the job
014040  000405                         BR      2$
                                ;
                                ;  Reenable output
                                ;
014042  042761  000000G 000000G 1$:    BIC     #$CTRLO,LSW3(R1);Remove output suppression
014050  004737  017216'               CALL    ECOCTL          ;Echo "^O"
                                ;
                                ;  Finished
                                ;
014054  000207                 2$:     RETURN
```

```
                        .SBTTL  ICPCTR -- Control-R processing
                ;-----------------------------------------------------------------------
                ;  Process control-R input character.
                ;
                ;  Inputs:
                ;   R1 = Virtual line index number.
                ;   R5 = Current input character.
                ;
014056  010246          ICPCTR: MOV     R2,-(SP)
014060  004737  000000G         CALL    BRKPT           ;Call TSEXEC for debugging breakpoint
014064  004737  017504'         CALL    SCACHR          ;Is job in single-char activation mode?
014070  103412                  BCS     9$              ;Br if yes
                ;
                ;  Redisplay the current input line
                ;
014072  004737  017216'  4$:     CALL    ECOCTL          ;Echo "^R"
014076  016102  000000G         MOV     LSTACT(R1),R2   ;Point past last activation char
014102  004737  016420'  1$:     CALL    FETCHR          ;Get next char from TT input buffer
014106  103403                  BCS     9$              ;Br if no more characters
014110  004737  017144'         CALL    ECHO1           ;Echo the character
014114  000772                  BR      1$              ;Continue printing line
                ;
                ;  Finished
                ;
014116  012602          9$:     MOV     (SP)+,R2
014120  000207                  RETURN
```

```
                                        .SBTTL  ICPCTU -- Control-U processing
                                ;-----------------------------------------------------------------------
                                ;  Process a Control-U character.
                                ;
                                ;  Inputs:
                                ;   R1 = Virtual line index number.
                                ;   R5 = Current input character.
                                ;
014122  010246                  ICPCTU: MOV     R2,-(SP)
014124  010346                          MOV     R3,-(SP)
                                ;
                                ;  Remember that last character was not escape or control-C
                                ;
014126  042761  000000G 000000G         BIC     #$1CTLC,LSW5(R1);Last character was not control-C
014134  042761  000000G 000000G         BIC     #$1ESC,LSW(R1)  ;Say last char was not escape
                                ;
                                ;  Determine if we are in single-character activation mode
                                ;
014142  004737  017504'                 CALL    SCACHR          ;See if we are in single-char activation mode
014146  103545                          BCS     27$             ;Br if yes
                                ;
                                ;  We are not in single-charactr activation mode
                                ;
014150  042761  000000G 000000G         BIC     #$RBOUT,LSW3(R1);Reset rubout mode
014156  032761  000000G 000000G         BIT     #$DODFR,LSW3(R1);Doing deferred echoing?
014164  001133                          BNE     5$              ;Br if yes
                                ;
                                ;  Determine if this is a scope type terminal
                                ;
014166  032761  000000G 000000G         BIT     #$SCOPE,LSW2(R1);Is this a scope terminal?
014174  001003                          BNE     15$             ;Br if yes
                                ;
                                ;  Echo "^U" for non-scope terminals
                                ;
014176  004737  017216'                 CALL    ECOCTL          ;Echo "^U"
014202  000524                          BR      5$              ;Go delete the characters
                                ;
                                ;  Do line erase for scope terminals
                                ;
014204  032761  000000G 000000G 15$:    BIT     #$1STCH,LSW3(R1);Any characters received yet?
014212  001523                          BEQ     27$             ;Br if not
014214  116103  000000G                 MOVB    LCOL(R1),R3     ;Get current column position
014220  042703  177400                  BIC     #^C377,R3       ;Kill sign extension
014224  116102  000000G                 MOVB    LINCUR(R1),R2   ;Get start of line position
014230  042702  177400                  BIC     #^C377,R2       ;Kill sign extension
014234  160203                          SUB     R2,R3           ;Get # of columns in field being erased
014236  003506                          BLE     5$              ;Br if nothing to erase
014240  126127  000000G 000000G         CMPB    LRBFIL(R1),#SPACE; Is rubout filler char = space?
014246  001056                          BNE     8$              ;Br if not
014250  032761  000000C 000000G         BIT     #VT52!VT100!VT2007!VT2008!HAZEL,LTRMTP(R1) ;Can we line erase?
014256  001452                          BEQ     8$              ;Br if not
014260  005702                          TST     R2              ;Are we erasing entire line?
014262  001005                          BNE     10$             ;Br if not
                                ;
                                ;  Going to start of line.  Use carriage-return to get there.
                                ;
014264  112700  000000G                 MOVB    #CR,R0          ;Send carriage return to terminal
```

```
     58 014270  004737  003156'                 CALL    PUTCH2
     59 014274  000405                          BR      11$
     60                                 ;
     61                                 ;  Not going to start of line.  Use backspaces to get to start of field.
     62                                 ;
     63 014276  112700  000000G         10$:    MOVB    #BKSPAC,R0      ;Get backspace character
     64 014302  004737  017144'         12$:    CALL    ECHO1           ;Send backspace to terminal
     65 014306  077303                          SOB     R3,12$          ;Send enough to get to start of field
     66                                 ;
     67                                 ;  Use scope control sequence to erase the line
     68                                 ;
     69 014310  012703  014470'         11$:    MOV     #ERV52,R3       ;Assume this is a VT52 terminal
     70 014314  116100  000000G                 MOVB    LNPRIM(R1),R0   ;Get primary job index number
     71 014320  032760  000000G 000000G        BIT     #$V52EM,LSW11(R0);Are we emulating a VT52?
     72 014326  001014                          BNE     13$             ;Br if yes
     73 014330  016100  000000G                 MOV     LTRMTP(R1),R0   ;Get terminal type code
     74 014334  032700  000000G                 BIT     #VT52,R0        ;Is this a VT52 terminal?
     75 014340  001007                          BNE     13$             ;Br if yes
     76 014342  012703  014477'                 MOV     #ERHAZL,R3      ;Assume hazeltine terminal
     77 014346  032700  000000G                 BIT     #HAZEL,R0       ;Is this a Hazeltine terminal?
     78 014352  001002                          BNE     13$             ;Br if yes
     79 014354  012703  014473'                 MOV     #ERV100,R3      ;Assume this is a VT100 or VT200
     80 014360  112300                  13$:    MOVB    (R3)+,R0        ;Get next char from control sequence
     81 014362  001434                          BEQ     5$              ;Br if end reached
     82 014364  120027  000037                  CMPB    R0,#37          ;Is this a control or printing character
     83 014370  101402                          BLOS    14$             ;Br if control character
     84 014372  105361  000000G                 DECB    LCOL(R1)        ;Correct column counter if sending printing ch
     85 014376  004737  003156'         14$:    CALL    PUTCH2          ;Send character to terminal
     86 014402  000766                          BR      13$             ;Continue sending control sequence
     87                                 ;
     88                                 ;  Kill all input characters by sending
     89                                 ;  backspace-space-backspace...
     90                                 ;
     91 014404  116102  000000G         8$:     MOVB    LRBFIL(R1),R2   ;Get rubout-filler character
     92 014410  032761  000000G 000000G        BIT     #$DBGMD,LSW6(R1);Is debugger doing I/O now?
     93 014416  001402                          BEQ     9$              ;Br if not
     94 014420  112702  000040                  MOVB    #' ,R2          ;Use blank for debugger rubout
     95 014424  112700  000000G         9$:     MOVB    #BKSPAC,R0
     96 014430  004737  017144'                 CALL    ECHO1           ;Send backspace
     97 014434  110200                          MOVB    R2,R0           ;Get rubout-filler character
     98 014436  004737  017144'                 CALL    ECHO1           ;Send filler character
     99 014442  112700  000000G                 MOVB    #BKSPAC,R0      ;Send backspace
    100 014446  004737  017144'                 CALL    ECHO1
    101 014452  077314                          SOB     R3,9$           ;Loop on number of characters to kill
    102                                 ;
    103                                 ;  Now delete characters from the input buffer
    104                                 ;
    105 014454  004737  015672'         5$:     CALL    KILCHR          ;Kill last character in the buffer
    106 014460  103375                          BCC     5$              ;Loop if more left to kill
    107                                 ;
    108                                 ;  Finished
    109                                 ;
    110 014462  012603                  27$:    MOV     (SP)+,R3
    111 014464  012602                          MOV     (SP)+,R2
    112 014466  000207                          RETURN
    113                                 ;
    114                                 ;  Terminal control sequences to erase a line.
```

```
    115                                         ;
    116 014470      000@    113     000  ERV52:   .BYTE   ESC,113,0          ;VT52
    117 014473      000@    133     113  ERV100:  .BYTE   ESC,133,113,0      ;VT100 and VT200
        014476      000
    118 014477      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.
                                ;
014502  032761  000000G 000000G ICPCTX: BIT     #SPCTTY,LJSW(R1);Is special character processing wanted?
014510  001415                          BEQ     2$              ;Br if not
014512  032761  000000G 000000G         BIT     #$CHACT,LSW5(R1);Is single character activation wanted?
014520  001403                          BEQ     1$              ;Br if not
014522  004737  016360'                 CALL    STRSNG          ;Store the character
014526  000410                          BR      9$
                                ;
                                ;  Special-TTY mode is set but not single character activation
                                ;
014530  010500                  1$:     MOV     R5,R0           ;Get char to echo
014532  004737  017216'                 CALL    ECOCTL          ;Echo"^X"
014536  004737  016300'                 CALL    STRACT          ;Store the character and activate
014542  000402                          BR      9$
                                ;
                                ;  Normal character processing
                                ;
014544  004737  012432'         2$:     CALL    REGCHR          ;Treat as regular character
                                ;
                                ;  Finished
                                ;
014550  042761  000000G 000000G 9$:     BIC     #$1CTLC,LSW5(R1);Say last character was not control-C
014556  042761  000000G 000000G         BIC     #$1ESC,LSW(R1)  ;Say last char was not escape
014564  000207                          RETURN
```

```
                                        .SBTTL  ICPCTZ -- Control-Z processing
                                ;-----------------------------------------------------------------------
                                ;  Process Control-Z character.
                                ;
                                ;  Inputs:
                                ;   R1 = Virtual line index number.
                                ;   R5 = Current input character.
                                ;
014566  004737  017504'         ICPCTZ: CALL    SCACHR          ;Are we in single character activation mode?
014572  103404                          BCS     9$              ;Br if yes
014574  004737  017216'                 CALL    ECOCTL          ;Echo "^Z"
014600  004737  016300'                 CALL    STRACT          ;Store character and activate
                                ;
                                ;  Say last character was not escape or control-C
                                ;
014604  042761  000000G 000000G 9$:     BIC     #$1CTLC,LSW5(R1);Say last character was not control-C
014612  042761  000000G 000000G         BIC     #$1ESC,LSW(R1)  ;Say last char was not escape
014620  000207                          RETURN
```

```
                                        .SBTTL  ICPESC -- Escape processing
                                ;-----------------------------------------------------------------------
                                ;  Process an escape character.
                                ;
                                ;  Inputs:
                                ;   R1 = Virtual line index number.
                                ;   R5 = Current input character.
                                ;
014622  004737  017362'         ICPESC: CALL    SLCHK           ;See if SL should get escape
014626  103410                          BCS     4$              ;Br if SL wants characters
014630  032761  000000G 000000G         BIT     #SPCTTY,LJSW(R1);Is job in special TTY mode?
014636  001407                          BEQ     1$              ;Br if not
014640  032761  000000G 000000G         BIT     #$CHACT,LSW5(R1);Is single char activation enabled?
014646  001412                          BEQ     2$              ;Br if not
014650  004737  016360'         4$:     CALL    STRSNG          ;Store and activate
014654  000432                          BR      9$
                                ;
                                ;  Escape and not single character activation
                                ;
014656  112700  000044          1$:     MOVB    #'$,R0          ;Echo "$"
014662  004737  017140'                  CALL    ECHO
014666  004737  016002'                  CALL    STRCHR          ;Store escape as non-activation char
014672  000423                          BR      9$
                                ;
                                ;  Program is in special-TTY mode but single character activation
                                ;  is not enabled.
                                ;
014674  112700  000044          2$:     MOVB    #'$,R0          ;Echo "$"
014700  004737  017140'                  CALL    ECHO
014704  032761  000000G 000000G         BIT     #$1ESC,LSW(R1)  ;Was last character ESC?
014712  001006                          BNE     3$              ;Br if yes
014714  004737  016002'                  CALL    STRCHR          ;Store the escape
014720  052761  000000G 000000G         BIS     #$1ESC,LSW(R1)  ;Remember that last char is escape
014726  000405                          BR      9$
014730  004737  016300'         3$:     CALL    STRACT          ;Store escape and activate
014734  042761  000000G 000000G         BIC     #$1ESC,LSW(R1)  ;Say last char was not escape
                                ;
                                ;  Finished
                                ;
014742  042761  000000G 000000G 9$:     BIC     #$1CTLC,LSW5(R1);Say last character was not control-C
014750  000207                          RETURN
```

```
                                        .SBTTL  ICPRUB -- Rubout processing
                                ;-----------------------------------------------------------------------
                                ;  Process a rubout character.
                                ;
                                ;  Inputs:
                                ;   R1 = Virtual line number.
                                ;   R5 = Rubout character.
                                ;
014752  010246                  ICPRUB: MOV     R2,-(SP)
014754  010346                          MOV     R3,-(SP)
                                ;
                                ;  Remember that last character was not control-C
                                ;
014756  042761  000000G 000000G         BIC     #$1CTLC,LSW5(R1);Last character was not control-C
014764  042761  000000G 000000G         BIC     #$1ESC,LSW(R1)  ;Say last char was not escape
                                ;
                                ;  See if we are in single-character activation mode
                                ;
014772  004737  017504'                 CALL    SCACHR          ;Are we in single-char activation mode
014776  103541                          BCS     2$              ;Br if yes
                                ;
                                ;  We are not in single character activation mode.
                                ;
015000  032761  000000G 000000G         BIT     #$ECHO,LSW2(R1) ;Is echoing turned on?
015006  001004                          BNE     12$             ;Br if yes
015010  032761  000000G 000000G         BIT     #$DBGMD,LSW6(R1);Is debugger in control?
015016  001417                          BEQ     30$             ;Br if not
015020  032761  000000G 000000G 12$:     BIT     #$SCOPE,LSW2(R1);Scope type terminal?
015026  001021                          BNE     27$             ;Br if yes
                                ;
                                ;  Rubout on non-scope terminal.
                                ;  Echo \xxx\ type rubout sequence.
                                ;
015030  032761  000000G 000000G         BIT     #$RBOUT,LSW3(R1);Are we already doing rubout sequence?
015036  001007                          BNE     30$             ;Br if yes
015040  112700  000134                  MOVB    #'\,R0          ;Get backslash to begin rubout sequence
015044  004737  017144'                 CALL    ECHO1           ;Echo "\"
015050  052761  000000G 000000G         BIS     #$RBOUT,LSW3(R1);Say we have started rubout sequence
015056  004737  015672'          30$:   CALL    KILCHR          ;Delete one character from buffer
015062  103507                          BCS     2$              ;Br if no char to delete
015064  004737  017144'                 CALL    ECHO1           ;Echo deleted char
015070  000504                          BR      2$
                                ;
                                ;  Do backspace editing for scope type terminals
                                ;
015072  004737  015672'          27$:   CALL    KILCHR          ;Delete one character from buffer
015076  103501                          BCS     2$              ;Br if no character was left to delete
015100  120027  000000G                 CMPB    R0,#TAB         ;Did we delete a tab character?
015104  001044                          BNE     10$
                                ;
                                ;  We just deleted a tab character.
                                ;  Do correct backspacing.
                                ;
015106  116103  000000G                 MOVB    LINCUR(R1),R3   ;Position of start of line
015112  042703  177400                  BIC     #^C377,R3       ;Kill sign extension
015116  016102  000000G                 MOV     LSTACT(R1),R2   ;Start of input string
015122  004737  016420'          3$:    CALL    FETCHR          ;Get next char from TT input buffer
```

```
     58 015126  103417                          BCS     6$              ;Br if no more characters
     59 015130  120027  000000G                 CMPB    R0,#TAB         ;Is character a tab?
     60 015134  001005                          BNE     7$              ;Br if not
     61 015136  062703  000010                  ADD     #8.,R3          ;Calculate spaces over it
     62 015142  042703  000007                  BIC     #7,R3
     63 015146  000765                          BR      3$
     64 015150  120027  000000G       7$:      CMPB    R0,#BKSPAC      ;Was character backspace?
     65 015154  001002                          BNE     5$              ;Br if not
     66 015156  005303                          DEC     R3              ;Backup cursor position
     67 015160  000760                          BR      3$
     68 015162  005203                  5$:     INC     R3              ;Advance cursor for normal character
     69 015164  000756                          BR      3$
     70 015166  010302                  6$:     MOV     R3,R2           ;Save position in front of last tab
     71 015170  062702  000010                  ADD     #8.,R2          ;Calc position after tab
     72 015174  042702  000007                  BIC     #7,R2
     73 015200  160302                          SUB     R3,R2           ;Calc number of backspace needed
     74 015202  012700  000000G                 MOV     #BKSPAC,R0      ;Now backspace cursor
     75 015206  004737  017144'          9$:     CALL    ECHO1           ;Backspace over tab columns
     76 015212  077203                          SOB     R2,9$
     77 015214  000432                          BR      2$
     78                                 ;
     79                                 ;  Check for rubout of backspace character
     80                                 ;
     81 015216  120027  000000G         10$:     CMPB    R0,#BKSPAC      ;Is deleted character a backspace?
     82 015222  001005                          BNE     11$             ;Br if not
     83 015224  112700  000000G                 MOVB    #SPACE,R0       ;Output a space to kill the backspace
     84 015230  004737  017144'                 CALL    ECHO1
     85 015234  000422                          BR      2$
     86                                 ;
     87                                 ;  Rubout of regular character.
     88                                 ;  Echo backspace-space-backspace
     89                                 ;
     90 015236  112700  000000G         11$:     MOVB    #BKSPAC,R0      ;Echo backspace
     91 015242  004737  017144'                 CALL    ECHO1
     92 015246  116100  000000G                 MOVB    LRBFIL(R1),R0   ;Get rubout-filler character
     93 015252  032761  000000G 000000G        BIT     #$DBGMD,LSW6(R1);Is a debugger running?
     94 015260  001402                          BEQ     13$             ;Br if not
     95 015262  112700  000040                  MOVB    #' ,R0          ;Use space for debugger rubout
     96 015266  004737  017144'          13$:    CALL    ECHO1           ;Echo rubout-filler character
     97 015272  112700  000000G                 MOVB    #BKSPAC,R0      ;Now echo backspace
     98 015276  004737  017144'                 CALL    ECHO1
     99                                 ;
    100                                 ;  Finished
    101                                 ;
    102 015302  012603                  2$:     MOV     (SP)+,R3
    103 015304  012602                          MOV     (SP)+,R2
    104 015306  000207                          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.
                                ;
015310                          CKVTAC:
                                ;
                                ;  See if this character is ESC, CSI, or SS3 which starts a terminal
                                ;  control sequence.
                                ;
015310  120527  000033                  CMPB    R5,#33          ;Is character ESC?
015314  001007                          BNE     1$              ;Br if not
015316  052761  000000G 000000G        BIS     #$1ESC,LSW(R1)  ;Remember last char was an escape
015324  052761  000000G 000000G        BIS     #$RTCS,LSW9(R1) ;Say we are receiving a control sequence
015332  000450                          BR      15$             ;Char is part of sequence
015334  120527  000233          1$:     CMPB    R5,#233         ;Is character CSI?
015340  001403                          BEQ     14$             ;Br if yes
015342  120527  000217                  CMPB    R5,#217         ;Is character SS3?
015346  001004                          BNE     2$              ;Br if not
                                ;
                                ;  This character begins a new control sequence
                                ;
015350  052761  000000G 000000G 14$:    BIS     #$RTCS,LSW9(R1) ;Say we are receiving a control sequence
015356  000433                          BR      3$              ;Go store the character
                                ;
                                ;  See if this character is part of a terminal control sequence
                                ;
015360  032761  000000G 000000G 2$:     BIT     #$RTCS,LSW9(R1) ;Are we currently receiving a control seq?
015366  001453                          BEQ     8$              ;Br if not
                                ;
                                ;  We are currently receiving a terminal control sequence.
                                ;  See if this character terminates the sequence.
                                ;
015370  120527  000101                  CMPB    R5,#'A          ;Is this a upper-case letter?
015374  103424                          BLO     3$              ;Br if not
015376  120527  000132                  CMPB    R5,#'Z
015402  101010                          BHI     7$              ;Br if not upper-case letter
015404  120527  000117                  CMPB    R5,#'O          ;Letter O?
015410  001013                          BNE     4$              ;Br if not
015412  032761  000000G 000000G        BIT     #$1ESC,LSW(R1)  ;Was last character ESC?
015420  001407                          BEQ     4$              ;Br if not
015422  000411                          BR      3$              ;ESC O is equivalent to SS3
015424  120527  000141          7$:     CMPB    R5,#141         ;Is this a lower-case letter?
015430  103406                          BLO     3$              ;Br if not
015432  120527  000176                  CMPB    R5,#176         ;Lower case letter or ~
015436  101003                          BHI     3$              ;Br if not
                                ;
                                ;  This character terminates the control sequence
                                ;
```

```
     58 015440  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 015446  042761  000000G 000000G 3$:     BIC     #$1ESC,LSW(R1)  ;Say last char was not escape
     63 015454  010546                 15$:    MOV     R5,-(SP)        ;Save the character
     64 015456  012705  000000G                MOV     #ESCFLG,R5      ;Get char that says escape sequence follows
     65 015462  004737  016002'                CALL    STRCHR          ;Store flag character
     66 015466  012605                         MOV     (SP)+,R5        ;Recover real character
     67 015470  032761  000000G 000000G       BIT     #$RTCS,LSW9(R1) ;Have we terminated the sequence?
     68 015476  001403                         BEQ     5$              ;Br if yes
     69 015500  004737  016002'                CALL    STRCHR          ;Store char that is part of sequence
     70 015504  000402                         BR      6$
     71 015506  004737  016300'          5$:    CALL    STRACT          ;Store char as activation character
     72 015512  000241                 6$:     CLC                     ;Say we finished processing the character
     73 015514  000401                         BR      9$
     74                                ;
     75                                ;  This character is not part of a control sequence
     76                                ;
     77 015516  000261                 8$:     SEC                     ;Signal char is not part of sequence
     78                                ;
     79                                ;  Finished
     80                                ;
     81 015520  000207                 9$:     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.
                                ;
015522  032761  000000G 000000G CHKODT: BIT     #$ODTMD,LSW4(R1);ODT ACTIVATION DESIRED?
015530  001417                         BEQ     1$              ;BRANCH IF NOT
                                ;
                                ;  User does want odt activation.
                                ;  Is this an ODT activation char?
                                ;
015532  120027  000054                 CMPB    R0,#',          ;DON'T ACTIVATE ON ','
015536  001414                         BEQ     1$
015540  120027  000044                 CMPB    R0,#'$          ;OR '$'
015544  001411                         BEQ     1$
015546  120027  000073                 CMPB    R0,#';          ;OR ';'
015552  001406                         BEQ     1$
015554  120027  000060                 CMPB    R0,#'0          ;IS THIS A DIGIT?
015560  103405                         BLO     2$              ;BRANCH IF NOT DIGIT
015562  120027  000071                 CMPB    R0,#'9
015566  101002                         BHI     2$              ;BRANCH IF NOT DIGIT
                                ;
                                ;  Not activation character.
                                ;
015570  000241                 1$:     CLC                     ;CLEAR C-FLAG
015572  000207                         RETURN
                                ;
                                ;  Got activation char.
                                ;
015574  000261                 2$:     SEC                     ;SET C-FLAG
015576  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.
                                ;
015600  010246                  INFIN:  MOV     R2,-(SP)
                                ;
                                ;  Determine which state to put job in
                                ;
015602  004737  017362'                 CALL    SLCHK           ;Is program doing single character activation?
015606  103003                          BCC     2$              ;Br if not
015610  012700  000000G                 MOV     #S$TTSC,R0      ;If yes, put job in special high-prio state
015614  000402                          BR      3$
015616  012700  000000G         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
                                ;
015622  016102  000000G         3$:     MOV     LSTATE(R1),R2   ;Get job's current execution state
015626  020200                          CMP     R2,R0           ;Is job already at that high a priority?
015630  101416                          BLOS    9$              ;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.
                                ;
015632  020227  000000G                 CMP     R2,#S$INWT      ;Is job waiting for TT input?
015636  001003                          BNE     4$              ;Br if not
015640  005061  000000G                 CLR     LRDTIM(R1)      ;Clear TT read time-out counter
015644  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.
                                ;
015646  020227  000000G         4$:     CMP     R2,#S$$RUN      ;Is job in a wait state?
015652  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.
                                ;
015654  005761  000000G                 TST     LITIME(R1)      ;Is job still interactive?
015660  001402                          BEQ     9$              ;Br if not
                                ;
                                ;  Boost priority of job
                                ;
015662  004737  000000G         1$:     CALL    ENQTL           ;Requeue job at tail of high-prio queue
                                ;
                                ;  Finished
                                ;
015666  012602                  9$:     MOV     (SP)+,R2
015670  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.
                ;
015672  010246  KILCHR: MOV     R2,-(SP)
015674  010346          MOV     R3,-(SP)
                ;
                ;  See if there are any characters in the TT input ring buffer
                ;
015676  005761  000000G         TST     LINCNT(R1)      ;ANY CHARS IN BUFFER NOW?
015702  001424          BEQ     3$              ;IF NOT NOTHING TO DELETE
                ;
                ;  Locate the last character in the TT input buffer
                ;
015704  016102  000000G         MOV     LINNXT(R1),R2   ;Get pointer past last char in buffer
015710  005302          DEC     R2              ;Point to last char in buffer
015712  020261  000000G         CMP     R2,LINBUF(R1)   ;Did we go past front of buffer?
015716  103003          BHIS    1$              ;Br if not
015720  016102  000000G         MOV     LINEND(R1),R2   ;Get pointer past right end of buffer
015724  005302          DEC     R2              ;Get pointer to last char in buffer
015726  010203  1$:     MOV     R2,R3           ;Save pointer to character being deleted
015730  004737  016420'         CALL    FETCHR          ;Get the last character in the buffer
                ;
                ;  Delete the last character unless it is an activation character
                ;
015734  032700  000000G         BIT     #ACFLAG,R0      ;Is this character an activation char?
015740  001005          BNE     3$              ;Br if yes
015742  010302          MOV     R3,R2           ;Get pointer to character to delete
015744  004737  016712'         CALL    DELCHR          ;Delete the character
                ;
                ;  We successfully deleted a character
                ;
015750  000241  2$:     CLC                     ;SAY CHAR WAS DELETED
015752  000401          BR      9$              ;Finished
                ;
                ;  There are no characters to delete
                ;
015754  000261  3$:     SEC                     ;Signal that there are no chars to delete
                ;
                ;  Finished
                ;
015756  012603  9$:     MOV     (SP)+,R3
015760  012602          MOV     (SP)+,R2
015762  000207          RETURN
```

```
                                        .SBTTL  INCHR  -- Store and echo a character
                                ;-----------------------------------------------------
                                ;  INCHR is called to store and echo the
                                ;  input character in R5.
                                ;  the user's index # must be in R1.
                                ;
015764  004737  016002'         INCHR:  CALL    STRCHR          ;STORE THE CHARACTER
015770  103403                          BCS     1$              ;BR IF INPUT BUFFER OVERFLOW
015772  010500                          MOV     R5,R0           ;GET CHARACTER INTO R0 FOR ECHO
015774  004737  017140'                 CALL    ECHO            ;ECHO THE CHARACTER
016000  000207                  1$:     RETURN
```

```
                                        .SBTTL  STRCHR -- Store a character into TT 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.
                                ;
016002  010046                  STRCHR: MOV     R0,-(SP)
016004  010246                          MOV     R2,-(SP)
016006  010346                          MOV     R3,-(SP)
                                ;
                                ;  Keep track of cursor position of 1st character on the line
                                ;
016010  032761  000000C 000000G         BIT     #<$1STCH!$DODFR>,LSW3(R1);1ST CHAR AFTER ACTIVATION?
016016  001006                          BNE     4$                      ;BRANCH IF NOT FIRST
016020  052761  000000G 000000G         BIS     #$1STCH,LSW3(R1);SAY WE'VE GOT A CHAR
016026  116161  000000G 000000G         MOVB    LCOL(R1),LINCUR(R1);SAVE CURSOR POSITION
                                ;
                                ;  See if buffer is about to overflow
                                ;
016034  016103  000000G         4$:     MOV     LINSPC(R1),R3   ;Get # free bytes in TT input ring buffer
016040  020327  000012                  CMP     R3,#10.         ;Got at least 10 free bytes?
016044  103022                          BHIS    7$              ;Br if yes
016046  005761  000000G                 TST     LACTIV(R1)      ;Gotten an activation char yet?
016052  001403                          BEQ     8$              ;Br if not
016054  052761  000000G 000000G         BIS     #$XSTOP,LSW6(R1);Don't move any more chars out of silo
016062  020327  000006         8$:     CMP     R3,#6.          ;Got at least 6 bytes left?
016066  103011                          BHIS    7$              ;Br if yes -- accept the character
016070  020327  000002                  CMP     R3,#2.          ;Is the buffer as full as we will allow?
016074  101470                          BLOS    5$              ;Br if yes -- Discard the character
016076  120527  000000G                 CMPB    R5,#CR          ;Is this a carriage return?
016102  001403                          BEQ     7$              ;Br if yes -- accept it
016104  032705  000000G                 BIT     #ACFLAG,R5      ;Is this an activation character?
016110  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.
                                ;
016112                          7$:     DISABL                  ;;;Disable interrupts
016120  016102  000000G                 MOV     LINNXT(R1),R2   ;;;Get pointer to next position in TT buffer
016124  010203                          MOV     R2,R3           ;;;
016126  005203                          INC     R3              ;;;Compute address of next char position
016130  020361  000000G                 CMP     R3,LINEND(R1)   ;;;Did we go past end of buffer?
016134  103402                          BLO     1$              ;;;Br if not
016136  016103  000000G                 MOV     LINBUF(R1),R3   ;;;Wrap around to front of buffer
016142  010361  000000G         1$:     MOV     R3,LINNXT(R1)   ;;;Save pointer to where next char goes
016146                                  ENABL                   ;Enable interrupts
                                ;
                                ;  Store character into buffer
                                ;
016154  010500                          MOV     R5,R0           ;Get character to store
016156  004737  016562'                 CALL    INSCHR          ;Store the character
                                ;
                                ;  Update character counts.
                                ;
016162  005361  000000G                 DEC     LINSPC(R1)      ;Reduce free space count for buffer
```

```
     58 016166  005261  000000G                 INC     LINCNT(R1)          ;Count another char in input buffer
     59                                 ;
     60                                 ;  Check for storing activation character.
     61                                 ;
     62 016172  032705  000000G                 BIT     #ACFLAG,R5          ;IS THIS AN ACTIVATION CHAR?
     63 016176  001414                          BEQ     3$                  ;BRANCH IF NOT
     64 016200  005261  000000G                 INC     LACTIV(R1)          ;Count another pending activ char
     65 016204  016161  000000G 000000G         MOV     LINNXT(R1),LSTACT(R1);REMEMBER POS OF CHAR
     66 016212  032761  000000G 000000G         BIT     #$DODFR,LSW3(R1);DOING DEFERED ECHOING?
     67 016220  001003                          BNE     3$                  ;BRANCH IF YES
     68 016222  042761  000000G 000000G         BIC     #$1STCH,LSW3(R1);SAY NO CHARS AFTER ACTIVATION
     69                                 ;
     70                                 ;  See if we need to trigger a completion routine for TT input
     71                                 ;
     72 016230  005761  000000G         3$:     TST     LTTCR(R1)           ;Does job want TT input compl routine?
     73 016234  001403                          BEQ     10$                 ;Br if not
     74 016236                                  OCALL   TTCPL               ;Trigger completion routine
     75                                 ;
     76                                 ;  Finished
     77                                 ;
     78 016244  000241                  10$:    CLC                         ;SIGNAL NO BUFFER OVERFLOW
     79 016246  012603                  6$:     MOV     (SP)+,R3
     80 016250  012602                          MOV     (SP)+,R2
     81 016252  012600                          MOV     (SP)+,R0
     82 016254  000207                          RETURN
     83                                 ;
     84                                 ;  Signal buffer overflow
     85                                 ;
     86 016256  052761  000000G 000000G 5$:     BIS     #$TTERR,LSW4(R1);REMEMBER THAT AN ERROR OCCURED
     87 016264  012700  000000G                 MOV     #BELL,R0            ;Get a bell character
     88 016270  004737  017200'                 CALL    ECHO2               ;Ring the bell to signal buffer nearly full
     89 016274  000261                          SEC                         ;SIGNAL OVERFLOW
     90 016276  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.
                                ;
016300  052705  000000G         STRACT: BIS     #ACFLAG,R5      ;SET ACTIVATION FLAG WITH CHAR
016304  004737  016002'                 CALL    STRCHR          ;STORE THE CHARACTER
016310  042705  000000G                 BIC     #ACFLAG,R5      ;RESET THE FLAG
016314  032761  000000G 000000G         BIT     #$DBGMD,LSW6(R1);IS DEBUGGER DOING I/O NOW?
016322  001004                         BNE     3$              ;BR IF YES - DON'T RESET ACTIVATION INFO
016324  005061  000000G                 CLR     LAFSIZ(R1)      ;RESET FIELD-WIDTH ACTIVATION
016330  005061  000000G                 CLR     LFWLIM(R1)      ;CLEAR FIELD WIDTH LIMIT
                                ;
                                ;  See if we should begin deferred char echoing.
                                ;
016334  032761  000000G 000000G 3$:     BIT     #$DEFER,LSW2(R1);DEFERRED MODE WANTED?
016342  001403                         BEQ     1$              ;BRANCH IF NOT
016344  052761  000000G 000000G         BIS     #$DODFR,LSW3(R1);BEGIN DEFERRED ECHOING
                                ;
                                ;  Activate the job
                                ;
016352  004737  015600'         1$:     CALL    INFIN           ;ACTIVATE THE USER
016356  000207                         RETURN
```

```
                                        .SBTTL  STRSNG -- Store char with single-character input
                                ;-----------------------------------------------------------------
                                ;  STRSNG is called to store a character received while in
                                ;  single character activation mode.
                                ;
                                ;  Inputs:
                                ;   R1 = Job index number.
                                ;   R5 = Character received.
                                ;
016360  010546                  STRSNG: MOV     R5,-(SP)        ;Save input character
                                ;
                                ;  Store the activation character
                                ;
016362  052705  000000G                 BIS     #ACFLAG,R5      ;Set activation flag with char
016366  004737  016002'                 CALL    STRCHR          ;Store the character
                                ;
                                ;  See if we need to begin deferred echo mode
                                ;
016372  032761  000000G 000000G         BIT     #$DEFER,LSW2(R1);Deferred mode wanted?
016400  001403                          BEQ     4$              ;Branch if not
016402  052761  000000G 000000G         BIS     #$DODFR,LSW3(R1);Begin deferred echoing
                                ;
                                ;  Activate the job
                                ;
016410  004737  015600'         4$:     CALL    INFIN           ;Activate the job
                                ;
                                ;  Finished
                                ;
016414  012605                  9$:     MOV     (SP)+,R5        ;Recover the original character
016416  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.
                                ;
016420  010446                  FETCHR: MOV     R4,-(SP)
016422  010546                          MOV     R5,-(SP)
                                ;
                                ;  See if there is another character in the ring buffer
                                ;
016424  020261  0000006                 CMP     R2,LINNXT(R1)   ;Are there any more chars in the buffer?
016430  001450                          BEQ     8$              ;Br if not
                                ;
                                ;  Compute index into bit vector that indicates if character is an
                                ;  activation character.
                                ;
016432  010204                          MOV     R2,R4           ;Get pointer to character
016434  166104  0000006                 SUB     LINBUF(R1),R4   ;Get character index into buffer
016440  073427  177775                  ASHC    #-3,R4          ;Get byte index into vector in R2
016444  072527  177763                  ASH     #-13.,R5        ;Right justify bit-within-byte index
016450  042705  177770                  BIC     #^C7,R5         ;Clear possible sign extension from shift
016454  066104  0000006                 ADD     LINEND(R1),R4   ;Get address of byte with flag bit
                                ;
                                ;  Get next character from TT ring buffer.
                                ;
016460                                  DISABL                  ;;;** Disable interrupts **
016466                                  TTMAP                   ;;;Map to TT buffer area
016502  005000                          CLR     R0              ;;;Initially clear R0
016504  152200                          BISB    (R2)+,R0        ;;;Get character without sign extension
016506  136514  000340'                 BITB    BITMSK(R5),(R4) ;;;Is this an activation character?
016512  001402                          BEQ     1$              ;;;Br if not
016514  052700  0000006                 BIS     #ACFLAG,R0      ;;;Remember this is an activation character
016520                          1$:     UNMAP                   ;;;Restore mapping
016526                                  ENABL                   ;** Enable interrupts **
                                ;
                                ;  See if we need to wrap around to front of ring buffer
                                ;
016534  020261  0000006         3$:     CMP     R2,LINEND(R1)   ;Did we just go past end of ring buffer?
016540  103402                          BLO     4$              ;Br if not
016542  016102  0000006                 MOV     LINBUF(R1),R2   ;Wrap around to front of ring buffer
                                ;
                                ;  We got a character
                                ;
016546  000241                  4$:     CLC                     ;Signal that we got a character
016550  000401                          BR      9$
                                ;
```

```
                                        ;  There are no more characters in the buffer
                                        ;
016552  000261                          8$:     SEC                     ;Signal that there are no more characters
                                        ;
                                        ;  Finished
                                        ;
016554  012605                          9$:     MOV     (SP)+,R5
016556  012604                                  MOV     (SP)+,R4
016560  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.
                                ;
016562  010446                  INSCHR: MOV     R4,-(SP)
016564  010546                          MOV     R5,-(SP)
                                ;
                                ;  Compute pointer into parallel bit vector with activation-character flags
                                ;
016566  010204                          MOV     R2,R4           ;Get character buffer pointer
016570  166104  000000G                 SUB     LINBUF(R1),R4   ;Compute byte index into buffer
016574  073427  177775                  ASHC    #-3,R4          ;Get byte index in R4
016600  072527  177763                  ASH     #-13.,R5        ;Right justify bit-within-byte index
016604  042705  177770                  BIC     #^C7,R5         ;Clear possible sign extension
016610  116505  000340'                 MOVB    BITMSK(R5),R5   ;Get bit mask
016614  066104  000000G                 ADD     LINEND(R1),R4   ;Get address of byte with activation flag
                                ;
                                ;  Store character into buffer and set or clear the activation-character flag
                                ;
016620                                  DISABL                  ;;;** Disable interrupts **
016626                                  TTMAP                   ;;;Map to TT buffer area
016642  110022                          MOVB    R0,(R2)+        ;;;Store character into buffer
016644  140514                          BICB    R5,(R4)         ;;;Clear the activation-character flag
016646  032700  000000G                 BIT     #ACFLAG,R0      ;;;Is this an activation character?
016652  001401                          BEQ     2$              ;;;Br if not
016654  150514                          BISB    R5,(R4)         ;;;Set the activation-character flag
016656                          2$:     UNMAP                   ;;;Restore mapping
016664                                  ENABL                   ;** Enable interrupts **
                                ;
                                ;  See if we need to wrap around buffer pointer
                                ;
016672  020261  000000G                 CMP     R2,LINEND(R1)   ;Do we need to wrap around to buffer front?
016676  103402                          BLO     9$              ;Br if not
016700  016102  000000G                 MOV     LINBUF(R1),R2   ;Wrap around to front of buffer
                                ;
                                ;  Finished
                                ;
016704  012605                  9$:     MOV     (SP)+,R5
016706  012604                          MOV     (SP)+,R4
016710  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.
                                ;
016712  010346                  DELCHR: MOV     R3,-(SP)
016714  010546                          MOV     R5,-(SP)
                                ;
                                ;  Get the character being deleted
                                ;
016716  010203                          MOV     R2,R3           ;Save original character pointer
016720  004737  016420'                 CALL    FETCHR          ;Get character being deleted
016724  103502                          BCS     20$             ;Br if no character to delete
                                ;
                                ;  See if we are deleting the 1st character in the buffer
                                ;
016726  020361  000000G                 CMP     R3,LINPNT(R1)   ;Is this the 1st char in the buffer?
016732  001003                          BNE     3$              ;Br if not
016734  010261  000000G                 MOV     R2,LINPNT(R1)   ;Set new pointer to 1st char in buffer
016740  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.
                                ;
016742  010046                  3$:     MOV     R0,-(SP)        ;Save character that is being deleted
016744  010346                          MOV     R3,-(SP)        ;Save pointer to deleted char position
016746                          4$:     DISABL                  ;;;** Disable interrupts **
016754  020261  000000G                 CMP     R2,LINNXT(R1)   ;;;Are we at the end of the chars now?
016760  001415                          BEQ     5$              ;;;Br if yes
016762                                  ENABL                   ;** Enable interrupts **
                                ;
                                ;  Slide characters over in buffer to fill in gap left by deleted character
                                ;
016770  010205                          MOV     R2,R5           ;Save pointer to following character
016772  004737  016420'                 CALL    FETCHR          ;Get next character in buffer
016776  010246                          MOV     R2,-(SP)        ;Save updated pointer
017000  010302                          MOV     R3,R2           ;Get pointer to pos to store character
017002  004737  016562'                 CALL    INSCHR          ;Reinsert the character
017006  010503                          MOV     R5,R3           ;Advance insert pointer
017010  012602                          MOV     (SP)+,R2        ;Get pointer to next char to move over
017012  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.
```

```
                                        ;
017014  010361  000000G         5$:    MOV     R3,LINNXT(R1)   ;;;Set new pointer to end of chars in buffer
017020                                  ENABL                   ;** Enable interrupts **
017026  012602                          MOV     (SP)+,R2        ;Get pointer to char following deleted one
017030  012600                          MOV     (SP)+,R0        ;Get character being deleted
                                        ;
                                        ;  Update character counters
                                        ;
017032  005261  000000G         8$:    INC     LINSPC(R1)      ;Another free char space in buffer
017036  005361  000000G                 DEC     LINCNT(R1)      ;One less char in buffer
017042  032700  000000G                 BIT     #ACFLAG,R0      ;Is deleted char an activation char?
017046  001402                          BEQ     19$             ;Br if not
017050  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.
                                        ;
017054  032761  000000G 000000G 19$:    BIT     #$XSTOP,LSW6(R1);Have we stopped input from silo buffer?
017062  001422                          BEQ     16$             ;Br if not
017064  026127  000000G 000017          CMP     LINCNT(R1),#15. ;Is input buffer almost empty?
017072  101406                          BLOS    17$             ;Br if yes
017074  032700  000000G                 BIT     #ACFLAG,R0      ;Is this an activation character?
017100  001413                          BEQ     16$             ;Br if not
017102  005761  000000G                 TST     LACTIV(R1)      ;Is this last activation char in buffer?
017106  001010                          BNE     16$             ;Br if not
017110  042761  000000G 000000G 17$:    BIC     #$XSTOP,LSW6(R1);Reenable input from silo
017116  052761  000000G 000000G         BIS     #$NDICP,LSW10(R1);Say line needs input character servicing
017124  005237  000000G                 INC     NEDCDI          ;Say input character processing needed
                                        ;
                                        ;  We deleted a character
                                        ;
017130  000241                  16$:    CLC                     ;Signal that we deleted a character
                                        ;
                                        ;  Finished
                                        ;
017132  012605                  20$:    MOV     (SP)+,R5
017134  012603                          MOV     (SP)+,R3
017136  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.
                                ;
017140  004737  017266'         ECHO:   CALL    RBEND           ;TERMINATE ANY RUBOUT FIELD
017144  004737  017522'         ECHO1:  CALL    CVTLC           ;CONVERT LOWER CASE CHARS TO UPPER CASE
017150  032761  000000G 000000G         BIT     #$DODFR,LSW3(R1);ARE WE DEFERRING ECHO NOW?
017156  001016                         BNE     ECHOR           ;BRANCH IF WE ARE
017160  032761  000000G 000000G         BIT     #$ECHO,LSW2(R1) ;IS CHARACTER ECHOING WANTED?
017166  001004                         BNE     ECHO2           ;BR IF YES
017170  032761  000000G 000000G         BIT     #$DBGMD,LSW6(R1);IS A DEBUGGER USING TERMINAL NOW?
017176  001406                         BEQ     ECHOR           ;BR IF NOT
017200  026127  000000G 000017  ECHO2:  CMP     LOTSPC(R1),#15. ;ROOM TO ECHO CHAR?
017206  002402                         BLT     ECHOR           ;BRANCH IF NOT
017210  004737  003156'                 CALL    PUTCH2          ;PUT CHAR IN OUTPUT BUFFER
017214  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.
                                ;
017216  010046                 ECOCTL: MOV     R0,-(SP)
017220  112700  000136                 MOVB    #136,R0         ;ECHO '^'
017224  004737  017140'                CALL    ECHO
017230  010500                         MOV     R5,R0           ;GET CONTROL CHARACTER
017232  052700  000100                 BIS     #100,R0         ;CONVERT TO PRINTING CHAR
017236  004737  017144'                CALL    ECHO1           ;PRINT CONTROL CHAR
017242  112700  000000G                MOVB    #CR,R0          ;PRINT CR-LF
017246  004737  017144'                CALL    ECHO1
017252  112700  000000G                MOVB    #LF,R0
017256  004737  017144'                CALL    ECHO1
017262  012600                         MOV     (SP)+,R0
017264  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.
                                ;
017266  032761  000000G 000000G RBEND:  BIT     #$RBOUT,LSW3(R1);ARE WE IN A RUBOUT FIELD?
017274  001411                         BEQ     1$              ;BRANCH IF NOT
017276  042761  000000G 000000G         BIC     #$RBOUT,LSW3(R1);CLEAR RUBOUT STATE
017304  010046                         MOV     R0,-(SP)
017306  112700  000134                 MOVB    #'\,R0          ;OUTPUT BACK SLASH
```

```
     58 017312  004737  017144'                    CALL     ECHO1
     59 017316  012600                             MOV      (SP)+,R0
     60 017320  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.
                                ;
017322  032761  000000G 000000G SCACHK: BIT     #SPCTTY,LJSW(R1);Does program want single char input?
017330  001412                          BEQ     1$              ;Br if not
017332  032761  000000G 000000G         BIT     #$CHACT,LSW5(R1);Is single character activation enabled?
017340  001406                          BEQ     1$              ;Br if not
017342  032761  000000G 000000G         BIT     #$DBGMD,LSW6(R1);Is a debugger using terminal now?
017350  001002                          BNE     1$              ;Br if yes
                                ;
                                ;  Job is in single character activation mode
                                ;
017352  000261                          SEC                     ;Signal that single char activation wanted
017354  000401                          BR      9$
                                ;
                                ;  Job does not want single char activation
                                ;
017356  000241                  1$:     CLC                     ;Signal no single char activation
                                ;
                                ;  Finished
                                ;
017360  000207                  9$:     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.
                                ;
017362  032761  000000G 000000G SLCHK:  BIT     #SPCTTY,LJSW(R1);Does program want single char input?
017370  001411                          BEQ     1$              ;Br if not
017372  032761  000000G 000000G        BIT     #$CHACT,LSW5(R1);Is single character activation enabled?
017400  001437                          BEQ     2$              ;Br if not
017402  032761  000000G 000000G        BIT     #$DBGMD,LSW6(R1);Is a debugger using terminal now?
017410  001033                          BNE     2$              ;Br if yes
017412  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.
                                ;
017414  032761  000000G 000000G 1$:     BIT     #$SLON,LSW7(R1) ;Is SL enabled for this line?
017422  001426                          BEQ     2$              ;Br if not
017424  032761  000000G 000000G        BIT     #DISSLE,LJSW(R1);Did program disable SL?
017432  001022                          BNE     2$              ;Br if yes
017434  032761  000000C 000000G        BIT     #<$ODTMD!$HITTY>,LSW4(R1);Are we in ODT or high efficiency?
017442  001016                          BNE     2$              ;Br if yes
017444  032761  000000G 000000G        BIT     #$VTESC,LSW5(R1);VTxxx activation enabled?
017452  001012                          BNE     2$              ;Br if yes
017454  032761  000000G 000000G        BIT     #$GTLIN,LSW4(R1);Is a .GTLIN being done?
017462  001004                          BNE     3$              ;Br if yes
017464  032761  000000G 000000G        BIT     #$SLTTY,LSW7(R1);Is SL enabled for TTY input?
017472  001402                          BEQ     2$              ;Br if not
                                ;
                                ;  We are in single line activation or single line editor mode.
                                ;
                                3$:     SEC                     ;Signal single character mode
017474  000261                          BR      9$
017476  000401                  ;
                                ;  We are not in single character mode.
                                ;
                                2$:     CLC                     ;Signal not single character mode
017500  000241                  ;
                                ;  Finished
                                ;
                                9$:     RETURN
017502  000207
```

```
                                        .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.
                                ;
017504  004737  017362'         SCACHR: CALL    SLCHK           ;Are we in single-char activation mode?
017510  103003                         BCC     9$              ;Br if not
                                ;
                                ;  We are in single-character activation mode.
                                ;  Pass the character to the program as an activation char.
                                ;
017512  004737  016360'                 CALL    STRSNG          ;Store character and activate
017516  000261                         SEC                     ;Say character was processed by us
                                ;
                                ;  Finished
                                ;
017520  000207                  9$:     RETURN
```

```
      1                                                 .SBTTL  CVTLC  -- Convert lower-case chars to upper-case
      2                                           ;-----------------------------------------------------------------
      3                                           ;  CVTLC is called to see if TT input characters entered
      4                                           ;  in lower case should be translated to upper case.
      5                                           ;  When called, R0 must contain the character to be tested.
      6                                           ;  R1 must contain the user line index #.
      7                                           ;  CVTLC checks to see if set-lc has been done and if the
      8                                           ;  bit is set in the JSW allowing lower case characters.
      9                                           ;  On return, the resulting character is returned in R0.
     10                                           ;  All other registers are preserved.
     11                                           ;
     12 017522  032761  000000G 000000G CVTLC:  BIT     #$LC,LSW2(R1)   ;WAS "SET TT LC" DONE?
     13 017530  001404                          BEQ     1$              ;BR IF NOT
     14 017532  032761  000000G 000000G         BIT     #LCBIT,LJSW(R1) ;IS LC-BIT SET IN JSW?
     15 017540  001014                          BNE     2$              ;BR IF YES (LC OK)
     16                                           ;
     17                                           ;  Translate lower case to upper case
     18                                           ;
     19 017542  010046                          1$:     MOV     R0,-(SP)        ;SAVE ORIGINAL CHARACTER VALUE
     20 017544  042700  000000G                 BIC     #ACFLAG,R0      ;CLEAR ACTIVATION FLAG
     21 017550  020027  000141                  CMP     R0,#141         ;LC('A')
     22 017554  002405                          BLT     3$              ;BR IF NOT LOWER-CASE LETTER
     23 017556  020027  000172                  CMP     R0,#172         ;LC('Z')
     24 017562  101002                          BHI     3$              ;BR IF NOT LOWER-CASE LETTER
     25 017564  042716  000040                  BIC     #40,(SP)        ;CONVERT LOWER-CASE LETTER TO UPPER-CASE
     26 017570  012600                          3$:     MOV     (SP)+,R0        ;GET POSSIBLY CONVERTED LETTER BACK TO R0
     27 017572  000207                          2$:     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.
                                ;
017574  010046                  SIGWAT: MOV     R0,-(SP)
017576  010146                          MOV     R1,-(SP)
                                ;
                                ;  Only signal if the job that is entering the wait state is not
                                ;  currently connected to the terminal.
                                ;
017600  016100  000000G                 MOV     LNPRIM(R1),R0   ;GET PRIMARY LINE #
017604  026001  000000G                 CMP     LNMAP(R0),R1    ;IS THE LINE CONNECTED TO TERM?
017610  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.
                                ;
017612  032761  000000G 000000G         BIT     #$VBELL,LSW9(R1);Have we already signaled wait state?
017620  001011                          BNE     9$              ;Br if yes
017622  052761  000000G 000000G         BIS     #$VBELL,LSW9(R1);Set flag saying we have signaled
                                ;
                                ;  Send bell to signal wait condition
                                ;
017630  016001  000000G                 MOV     LNMAP(R0),R1    ;GET CURRENTLY CONNECTED LINE #
017634  112700  000000G                 MOVB    #BELL,R0        ;SEND BELL AS SIGNAL CHARACTER
017640  004737  005756'                 CALL    TRYCHR
                                ;
                                ;  Finished
                                ;
017644  012601                  9$:     MOV     (SP)+,R1
017646  012600                          MOV     (SP)+,R0
017650  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.
                                ;
017652  010446                  SIGBRK: MOV     R4,-(SP)
017654  016104  000000G                 MOV     LBRKCQ(R1),R4   ;DOES USER WANT NOTIFICATION OF BREAK?
017660  001404                          BEQ     1$              ;BR IF NOT
017662  005061  000000G                 CLR     LBRKCQ(R1)      ;SAY THAT BREAK QUEUE ELEMENT HAS BEEN USED UP
017666  004737  000000G                 CALL    QCOMPL          ;QUEUE COMPLETION ROUTINE FOR THE JOB
017672  012604                  1$:     MOV     (SP)+,R4
017674  000207                          RETURN
```

```
     58         000001                              .END
Errors detected:   0

*** Assembler statistics


Work  file  reads:  0
Work  file writes:  0
Size of work file:  10432 Words  ( 41 Pages)
Size of core pool:  18176 Words  ( 71 Pages)
Operating  system:  RT-11

Elapsed time:  00:01:17.17
,LP:TSTTY=DK:TSTTY/C/N:SYM
```

Cross reference table (CREF V05.05)

```
$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    50-79   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
$ALTER    1-95     9-129   9-139
$AUTO     1-34    49-32
$CCLRN    1-76    35-41
$CFABT    1-86     9-120   9-122
$CFALL    1-51    39-75   39-82   41-17   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    50-77   50-80   50-85   50-87   50-91
$DBGBK    1-56    56-17
$DBGMD    1-82     7-22   35-32   38-25   41-13   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   35-51   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    9-156  33-49   41-15   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    9-143  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    52-23
$QUIET    1-50     9-104  35-21   35-49   36-16   39-53   39-61   39-68
```

Cross reference table (CREF V05.05)

```
$RBOUT    1-48     60-24    64-34    64-38    76-53    76-55
$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     35-38    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    43-132*
CFPNT     1-61      9-100    9-111    9-118    9-127    13-29    25-71    40-31    40-45    40-52*   41-10    42-10*
         42-33*    43-13
CFPOP    40-44     43-6#
CFPSAV    1-61
CFSP      1-91     43-60    43-131*
```

```
CFSPND    1-69      9-48*     9-100*    9-152     9-155*    25-73*    25-81     25-84*
CFSTOP    9-101     25-74     42-6#     43-34
CFSTRT    9-154     25-83     42-29#    43-116
CFTEST    7-15      27-16     29-18     35-4      35-43     35-47     36-14     39-14     41-10#
CFTNO    41-11      41-14     41-18     41-21#
CFTST1   41-12      41-15#
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     50-69     50-75#
CKHIIN   50-50      50-57#
CKICTL   50-88      50-90     50-101#
CKSPAC   50-34      50-36     50-42#
CKVTAC   50-70      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       9-132    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   52-40      52-49#
CTRLC     1-64       9-73      9-74      24-22     26-33     35-30     38-47     49-39
```

Cross reference table (CREF V05.05)

```
CTRLX     1-64      38-49
CTRLZ     1-64       1-89      3-24      9-69      9-141     38-45
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
DOCTLT    1-38      52-16
DOCTRL   50-103     52-10#
DOSCHD    1-70       2-33      2-54      4-17     11-26     55-46*
DOSWIT    1-36      50-96
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     35-44     36-7      36-12#
GCEXIT   33-54      36-13     36-17     36-19     36-21#
```

Cross reference table (CREF V05.05)

```
GCH1    33-14#  33-65   33-68   35-40
GETCHR   3-23    7-35    8-42    9-121   9-131   33-10#
GETUCH   1-88    2-41    8-22    8-26
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   52-62   53-9#   54-36
ICPCTC  52-52   55-9#
ICPCTD  52-53   56-10#
ICPCTG  52-56   57-9#
ICPCTO  52-64   58-9#
ICPCTR  52-67   59-9#
ICPCTU  52-70   60-9#
ICPCTX  52-73   61-9#
ICPCTZ  52-75   62-9#
ICPESC  52-76   63-9#
ICPLF   52-59   54-9#
ICPRUB  50-107  64-9#
ILSW2    1-44   49-32
ILWAIT  33-64   46-7#
IN$ACT   1-42    9-61   43-143
INCHR   50-83   51-71   53-32   69-7#
INFIN   67-10#  71-28   72-25
INITFL   1-71   49-13
INITLN   1-92   49-45
INSCHR  24-47   27-44   70-53   74-15#  75-51
INTPRI   1-69    1-70    1-79   18-51   18-57   19-47   46-44   47-20   47-58   70-48   73-45   74-38
        75-43   75-60
JSWLOC   1-71   25-32   25-35*
KILCHR  60-105  64-39   64-46   68-13#
KPAR6    1-86   18-41   18-41*  18-43*  19-40   19-40*  19-42*  73-38   73-38*  73-44*  74-31   74-31*
        74-37*
LABTIM   1-45   49-51
LACTIV   1-61    7-13    8-45   24-48*  27-20   27-45*  33-62   46-33   47-18   55-69*  70-25   70-64*
        75-70*  75-81
LAFSIZ   1-69   24-49*  27-46*  27-53*  51-44   55-59*  71-17*
LBRKCH   1-80   50-33
LBRKCQ   1-82   81-52   81-54*
LCBIT    1-68    1-79   25-33   55-51   80-14
LCOL     1-72   14-41*  16-13*  16-15*  16-21*  16-28   16-40*  47-26   60-42   60-84*  70-18
LESCHR   1-53   20-28   20-34   34-40*
LESRTN   1-53   20-18   34-34*
LF       1-67    9-93    9-123  16-50   20-46   20-54   24-61   37-9    38-21   38-60   44-36   53-16
        53-26   53-33   76-40
LF$IN    1-95   35-45   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     35-53     36-20     44-9#
LOGCR     1-23     35-54     44-33#
LOGEND    1-94     45-20
LOGFLG    1-95     12-69     35-45     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     50-79*    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*   52-23     60-71
LSW2      1-41      9-129     9-139     12-31     16-29     16-46     25-5*     25-12*    25-24*    25-31*    25-42*    25-49*
         25-56*    30-5*     30-12*    35-51     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     50-77     50-80*    50-85*    50-87     50-91*    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     9-143     9-156*    12-16     13-27     24-24*    26-35*
```

```
PUTUCH    1-88      3-28      3-45      8-43
PVNPW     1-36      43-46     43-110
QCOMPL    1-47      81-55
QHDSPN    1-74      31-14     46-41
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#
REQCHR    50-109    51-9#     52-49     52-50     52-51     52-54     52-55     52-57     52-58     52-60     52-61     52-63
          52-65     52-66     52-68     52-69     52-71     52-72     52-74     52-77     52-78     52-79     52-80     56-22
          57-25     61-25
RSSPAC    1-29      26-5      26-10#
RUBOUT    1-66      14-39     50-105
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    50-81     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      9-145     12-26     19-19     33-16     33-30     33-37     35-55     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#
```

Cross reference table (CREF V05.05)

```
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*
VALADB    1-88      2-8       2-14      3-8       3-13      5-9       8-17      8-36      8-40
VCTRLT    1-38     52-14
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     50-75
VVPWCH    1-36     52-21
WINCHR    1-52     17-19      19-25
WINPRT    1-35     52-32
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 V05.05)

```
...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     50-96     52-16     52-32     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#
```