Table of contents

```
                                        .TITLE  TSCLO -- Communication Line (CL) Handler for TSX-Plus
000000                                  .PSECT  TSCLO
                                        .ENABL  LC
                                        .ENABL  AMA
                                        .DSABL  GBL
000000  012257                          .RAD50  /CLO/           ;Virtual segment ID word
                                ;-------------------------------------------------------------------
                                ;  TSCLO is a system virtual overlay which provides support for the
                                ;  Communication Line (CL) handler for TSX-Plus.
                                ;  This handler supports I/O to communication lines declared by
                                ;  use of the IOLINE macro when the system is generated.
                                ;  The device names are CL0, CL1, ..., CL7, C10, C11, ..., C17.
                                ;  Internal queueing is used to allow concurrent input/output operations
                                ;  to take place on all of the devices at the same time.
                                ;  XON/XOFF support is provided.
                                ;
                                ;  Copyright (c) 1984, 1985.
                                ;  S&H Computer Systems, Inc.
                                ;  Nashville, Tennessee  USA
                                ;  All rights reserved
                                ;
                                ;
                                ;  Global definitions
                                ;
                                        .GLOBL  CLIOQ,CLABRT,CLTIMR,CLINCP,CLXICP,CLXBRK
                                        .GLOBL  SETDTR,CLREST
                                ;
                                ;  Global references
                                ;
                                        .GLOBL  GETRTQ,CQ$LOT,CQ$RTN,KPAR5,CQ$PA5,Q.DEVX
                                        .GLOBL  CQ$RO,MRKTIM,CQ$LNK,VCXTRM,VCXCTL,CL$ORG
                                        .GLOBL  CLTOTL,LSW3,Q.JOB,LXCL,CL$XLN,C1DEVX
                                        .GLOBL  PSW,INTPRI,PTWRD,$XCHAR,TRNSTR
                                        .GLOBL  Q.WCNT,Q.BLKN,Q.LINK,IOFIN,CM$MCC,CM$FFI
                                        .GLOBL  $CTRLS,LSW3,SETSPD,CDSXON,CM$WRT
                                        .GLOBL  TTINCP,LINIR,FORCEX,LNMAP
                                        .GLOBL  CL$EPN,CL$EPS,CL$EPP,CM$EFP,CLEOFS
                                        .GLOBL  CDSTRT,LCDTYP,PTBYT,CLVERS,CLSFWB
                                        .GLOBL  LHIRBS,$HISTP,CLSFAB,NEDCDO,NEDCLO
                                        .GLOBL  LHIRBB,LHIRBQ,LHIRBP,LHIRBA
                                        .GLOBL  Q.UNIT,Q.FUNC,Q.CSW,CS$ERR,C.CSW
                                        .GLOBL  CO$FF,CO$TAB,CO$LFO,CO$LFI,CO$FFO
                                        .GLOBL  CO$BNO,CO$BNI,LSW1O,CO$8BT
                                        .GLOBL  CM$TBS,CM$IRQ,CM$ON,CM$EOF
                                        .GLOBL  CL$OPT,CL$STA,SILFET,CL$ORA,CLSFIC,CLSFOC
                                        .GLOBL  FRKGET,FORKQ,FQ$PRI,FQ$RTN,FP$IOA
                                        .GLOBL  CL$COL,CL$RQH,CL$WQH,CLABF
                                        .GLOBL  CLCQE,CLLQE,CM$ORP,CL$ORS,CL$ORP
                                        .GLOBL  CL$ORS,CL$ORE,CL$ORB,QTBYT,CLSFMS
                                        .GLOBL  CS$EOF,CO$DTR,CM$DTR,CLSFRL
                                        .GLOBL  CM$FFS,CL$LIN,CL$LEN,CO$LC,CL$WID
                                        .GLOBL  CO$CTL,CL$SKP,CO$CR,KPAR6,Q.PAR,Q.BUFF
                                        .GLOBL  CLSFCH,CLSFBC,CLSFRB,CLSFHS,CLSFDL
                                        .GLOBL  CLSFSO,CLSFCO,CLSFSL,CLSFSS,CLSFSW
                                        .GLOBL  MS$DTR,CDSDSS,CDQDSS,OVRHC,LCDTYP
                                        .GLOBL  CM$CRL,CDQDSS,MS$CAR,MS$RNG
                                        .GLOBL  CM$BRK,CDSBRK,MS$BRK,CLSFSP
```

```
                                        .GLOBL  CL$LIX,LCLUNT
                                        .GLOBL  GETDSS,SETDSS,XL$XFX,XL$XFR,XL$CTS,XL$CD,XL$RI
```

```
                        ;
                        ;-------------------------------------------------------------------------------
                        ;  Macro definitions
                        ;
                        ;  Disable interrupts
                        ;
                              .MACRO  DISABL            ;Disable interrupts
                              BIS     #340,@#PSW
                              .ENDM   DISABL
                        ;
                        ;  Enable interrupts
                        ;
                              .MACRO  ENABL             ;Enable interrupts
                              BIC     INTPRI,@#PSW
                              .ENDM   ENABL
                        ;
                        ;  Call another system virtual overlay region
                        ;
                              .MACRO  OCALL   ENTADD
                              CALL    OVRHC
                              .WORD   ENTADD
                              .ENDM   OCALL
```

```
                                .SBTTL  Parameter definitions
                        ;-----------------------------------------------------------------------
                        ;  Ascii characters
                        ;
000015                  CR      =       15              ;Carriage return
000012                  LF      =       12              ;Line feed
000014                  FF      =       14              ;Form feed
000023                  CTRLS   =       23              ;Ctrl-S
000021                  CTRLQ   =       21              ;Ctrl-Q
000032                  CTRLZ   =       32              ;Ctrl-Z
000040                  SPACE   =       40              ;Space
```

```
                                        .SBTTL  Data areas
                                ;-------------------------------------------------------------------------------
                                ;  General data areas
                                ;
000002  177777                 RTNCNT: .WORD   -1              ;Counts if someone in RTNQ routine
000004  000000                 CQH:    .WORD   0               ;List head for Q elements waiting to be freed
000006  000000                 ABTQFL: .WORD   0               ;non-zero ==> RTNQ fork request pending
```

```
                                        .SBTTL  CLENTR -- Entry point for processing a new I/O request
                                ;-----------------------------------------------------------------------
                                ;  CLIOQ is called by the system I/O initiation routine to start a new
                                ;  I/O request.
                                ;  We process some requests immediately, but for most (such as read and
                                ;  write) we move the request from the handler queue onto an internal
                                ;  queue.
                                ;
                                ;  Inputs:
                                ;   CLCQE = Current queue request.
                                ;   CLLQE = Last queue request.
                                ;
000010  010346                  CLIOQ:  MOV     R3,-(SP)
000012  010446                          MOV     R4,-(SP)
000014  010546                          MOV     R5,-(SP)
                                ;
                                ;  Remove current queue element from list pointed to by handler header
                                ;
000016                          CLQOK:  DISABL                          ;;;** Disable interrupts **
000024  013704  0000000                 MOV     CLCQE,R4                ;;;Get pointer to queue element
000030  001406                          BEQ     1$                      ;;;Br if there is no queue element to process
000032  016437  000000C 0000000         MOV     Q.LINK-Q.BLKN(R4),CLCQE;;;Remove queue element from list
000040  001002                          BNE     1$                      ;;;Br if more elements pending
000042  005037  0000000                 CLR     CLLQE                   ;;;Say there are no pending queue elements
000046                          1$:     ENABL                           ;** Enable interrupts **
000054  005704                          TST     R4                      ;Is there a queue element to process?
000056  001004                          BNE     3$                      ;Br if yes
                                ;
                                ;  There are no remaining queue elements for the handler to process.
                                ;  Return to the system.
                                ;
000060  012605                          MOV     (SP)+,R5
000062  012604                          MOV     (SP)+,R4
000064  012603                          MOV     (SP)+,R3
000066  000207                          RETURN
                                ;
                                ;  There is a queue request to be processed.
                                ;  R4 = Points to Q.BLKN cell in queue element.
                                ;  Determine if I/O is being done to a valid CL unit
                                ;
000070  116405  000000C         3$:     MOVB    Q.UNIT-Q.BLKN(R4),R5;Get device unit number
000074  042705  177770                  BIC     #^C7,R5                 ;Clear all but unit # field
000100  126437  000000C 0000000         CMPB    Q.DEVX-Q.BLKN(R4),C1DEVX ;Is the a C1 unit?
000106  001002                          BNE     4$                      ;Br if not
000110  062705  000010                  ADD     #8.,R5                  ;Bias C1 unit numbers by 8
000114  006305                  4$:     ASL     R5                      ;Convert to word index
000116  016501  0000000                 MOV     CL$LIX(R5),R1           ;Is this CL unit associated with a line?
000122  001002                          BNE     2$                      ;Br if yes -- This is a valid CL unit
000124  000137  000322'                 JMP     CLERR                   ;Return immediate hard error code
                                ;
                                ;  Get the function code and see if this is a .READ, .WRITE, or .SPFUN.
                                ;  R5 = CL unit index number.
                                ;
000130  116403  000000C         2$:     MOVB    Q.FUNC-Q.BLKN(R4),R3 ;Get the function code
000134  001037                          BNE     CLSPFN                  ;Br if this is a .SPFUN operation
000136  006364  000000C                 ASL     Q.WCNT-Q.BLKN(R4) ;Convert word count to # bytes
000142  103415                          BCS     CLWRIT                  ;Br if this is a write operation
```

```
000144  001002                          BNE     CLREAD          ;Br if this is a read operation
000146  000137  001534'                 JMP     CLQXIT          ;Br if this is a seek operation
                                ;
                                ;  This is a .READ operation.
                                ;  Move queue entry to internal read queue for this unit.
                                ;
000152  004737  006220'         CLREAD: CALL    LINON           ;Turn on the line
000156  012703  000000G                 MOV     #CL$RQH,R3      ;Get pointer to read queue head
000162  004737  005724'                 CALL    MOVQ            ;Move queue element to internal queue
                                ;
                                ;  Call routine to move any pending characters in silo buffer for this
                                ;  line into the data buffer.
                                ;
000166  004737  002204'                 CALL    IRINGQ          ;Move chars from silo buffer to data buffer
                                ;
                                ;  Finished starting the read operation.
                                ;  The queue element will be returned to the system when we have
                                ;  completed the operation.
                                ;
000172  000137  000016'                 JMP     CLQOK
                                ;
                                ;  This is a .WRITE request.
                                ;  Move queue element to internal write queue for this CL unit.
                                ;
000176  005464  000000C         CLWRIT: NEG     Q.WCNT-Q.BLKN(R4) ;Make write byte count positive
000202  052765  000000G 000000G CLWRTB: BIS     #CM$WRT,CL$STA(R5);Set flag that says a write has been done
000210  012703  000000G                 MOV     #CL$WQH,R3      ;Get pointer to write queue head
000214  004737  005724'                 CALL    MOVQ            ;Move queue element to write queue
000220  004737  006220'                 CALL    LINON           ;Turn on the line
                                ;
                                ;  Move characters from data buffer to output ring buffer and then
                                ;  start output to the line.
                                ;
000224  004737  003122'                 CALL    ORINGP          ;Move chars from data buffer to ring buffer
                                ;
                                ;  Finished starting a write operation
                                ;
000230  000137  000016'                 JMP     CLQOK
```

```
                                        .SBTTL  CLSPFN -- .SPFUN processing
                                ;-----------------------------------------------------------------------
                                ;  The current queue request is for a .SPFUN operation
                                ;  At this point the following registers are set up:
                                ;   R1 = TSX-Plus line index number of line being used by CL unit.
                                ;   R3 = .SPFUN code from Q.FUNC.
                                ;   R4 = Pointer to Q.BLKN field of current queue element
                                ;   R5 = CL unit index number
                                ;
000234                          CLSPFN:
                                ;
                                ;  See which group of special functions this code is in
                                ;
000234  042703  177400                  BIC     #^C<377>,R3     ;Clear sign extension
000240  001430                          BEQ     CLERR           ;Function code of 0 is invalid
000242  020327  000004                  CMP     R3,#MAXSF0      ;Too big for group 0?
000246  101420                          BLOS    3$              ;Br if in group 0
000250  020327  000201                  CMP     R3,#201         ;Is this code too small?
000254  103422                          BLO     CLERR           ;Br if too small
000256  020327  000206                  CMP     R3,#MAXSF1      ;Is it in group 1?
000262  101410                          BLOS    2$              ;Br if yes
000264  020327  000250                  CMP     R3,#250         ;Is it in group 2?
000270  103414                          BLO     CLERR           ;Br if too small for group 2
000272  020327  000266                  CMP     R3,#MAXSF2      ;Is it within group 2?
000276  101011                          BHI     CLERR           ;Br if not
000300  162703  000041                  SUB     #247-MAXSF1,R3  ;Correct for group 1 codes
000304  162703  000174          2$:     SUB     #200-MAXSF0,R3  ;Correct for group 0 codes
                                ;
                                ;  Branch off to processing routine
                                ;
000310  162703  000001          3$:     SUB     #1,R3           ;Subtract lowest function code
000314  006303                          ASL     R3              ;Convert function code to word table index
000316  000173  000340'                 JMP     @SPFRTN(R3)     ;Enter processing routine
                                ;
                                ;  Invalid special function code
                                ;
000322  016400  000000C         CLERR:  MOV     Q.CSW-Q.BLKN(R4),R0 ;Get address of CSW
000326  052760  000000G 000000G         BIS     #CS$ERR,C.CSW(R0) ;Set hard error flag in CSW
000334  000137  001534'                 JMP     CLQXIT          ;Do .DRFIN to tell system this op is completed
```

```
                                ;-----------------------------------------------------------------------
                                ;  Branch vector for .SPFUN processing routines based on function code value.
                                ;
                                SPFRTN:
                                ;
                                ;  Group 0: Function codes in the range 1 to 4
                                ;
 000340  000422'        SFGRP0: .WORD   SFCLOS           ;001 - Close file
 000342  000604'                .WORD   SFTERM           ;002 - Delete file
 000344  001534'                .WORD   CLQXIT           ;003 - Lookup file
 000346  001534'                .WORD   CLQXIT           ;004 - Enter file
         000004         MAXSF0  =       <.-SFGRP0>/2     ;Maximum function code value in group 0
                                ;
                                ;  Group 1: Function codes in the range 201 to 247.
                                ;
 000350  000432'        SFGRP1: .WORD   SFCLER           ;201 - Clear flags
 000352  000474'                .WORD   SFBREK           ;202 - Break transmission control
 000354  000552'                .WORD   SFREAD           ;203 - Special read with byte count
 000356  000564'                .WORD   SFSTAT           ;204 - Get handler status
 000360  000604'                .WORD   SFTERM           ;205 - Terminate I/O
 000362  000634'                .WORD   SFDTR            ;206 - Raise or drop DTR signal
         000206         MAXSF1  =       200+<<.-SFGRP1>/2> ;Highest function code in group 1
                                ;
                                ;  Group 2: Function codes with values of 250 or greater.
                                ;
 000364  000670'        SFGRP2: .WORD   SFSOPT           ;250 - Set option flags
 000366  000710'                .WORD   SFCOPT           ;251 - Clear option flags
 000370  000730'                .WORD   SFSLEN           ;252 - Set page length
 000372  000744'                .WORD   SFSSKP           ;253 - Set skip lines
 000374  000760'                .WORD   SFSWID           ;254 - Set page width
 000376  000774'                .WORD   SFGMS            ;255 - Get modem status
 000400  001014'                .WORD   SFSPD            ;256 - Set transmit/receive speed
 000402  001032'                .WORD   SFABT            ;257 - Abort all pending read/write requests
 000404  000152'                .WORD   CLREAD           ;260 - Read line with byte count
 000406  001072'                .WORD   SFIC             ;261 - Get number of pending input characters
 000410  001112'                .WORD   SFOC             ;262 - Get number of pending output chars
 000412  000202'                .WORD   CLWRTB           ;263 - Write with byte count
 000414  001170'                .WORD   SFSEFP           ;264 - Set end-of-file output control
 000416  001246'                .WORD   SFREST           ;265 - Reset CL unit
 000420  001256'                .WORD   SFGOPT           ;266 - Get current options and settings
         000266         MAXSF2  =       247+<<.-SFGRP2>/2> ;Highest legal function # in group 2
```

```
                                ;-----------------------------------------------------------------------
                                ;  Special function # 1
                                ;  Close file.
                                ;
000422  004737  001712'        SFCLOS: CALL    CLCLOS          ;Perform end-of-file operations
000426  000137  001534'                JMP     CLQXIT          ;Finished
                                ;
                                ;-----------------------------------------------------------------------
                                ;  Special function # 201
                                ;  Clear handler flags.
                                ;  The effect is to clear the flag saying we have received an XOFF
                                ;  and to send an XON.
                                ;
000432                          SFCLER:
                                ;
                                ;  Clear flag saying we have received an XOFF
                                ;
000432  042761  000000G 000000G         BIC     #$CTRLS,LSW3(R1);Clear the ctrl-S flag
                                ;
                                ;  Send an XON
                                ;
000440  042761  000000G 000000G         BIC     #$HISTP,LSW10(R1);Say input has not been stopped by XOFF
000446  016100  000000G                 MOV     LCDTYP(R1),R0   ;Get device type code
000452  004770  000000G                 CALL    @CDSXON(R0)     ;Call routine to stuff XON into output
                                ;
                                ;  Start output
                                ;
000456  004737  005506'                 CALL    CLSTRT          ;Start transmission
                                ;
                                ;  Clear end of file flag
                                ;
000462  042765  0000000 0000000         BIC     #CM$EOF,CL$STA(R5);Clear end of file status
                                ;
                                ;  Finished
                                ;
000470  000137  001534'                 JMP     CLQXIT          ;Finished with operation
                                ;
                                ;-----------------------------------------------------------------------
                                ;  Special function # 202
                                ;  Start or stop sending a break.
                                ;  Word count non-zero ==> Start sending a break.
                                ;  Word count zero     ==> End sending a break.
                                ;
                                ;
000474  005764  000000C         SFBREK: TST     Q.WCNT-Q.BLKN(R4) ;Start or end break?
000500  001412                         BEQ     1$              ;Br if we are ending a break
                                ;
                                ;  Begin sending a break
                                ;
000502  052765  0000000 0000000         BIS     #CM$BRK,CL$STA(R5);Set flag saying we are sending a break
000510  012700  000000G                 MOV     #MS$BRK,R0      ;Set flag to start break transmission
000514  004737  006340'                 CALL    SETBRK          ;Call hardware routine to start sending break
000520  004737  005506'                 CALL    CLSTRT          ;Start transmitter
000524  000410                         BR      9$
                                ;
                                ;  End sending a break
                                ;
```

```
                        1$:     CLR     R0              ;Clear break-send flag
                                CALL    SETBRK          ;Call hardware routine to end break
                                BIC     #CM$BRK,CL$STA(R5);Clear flag that says we are sending a break
                                CALL    CLSTRT          ;Start transmitter
                        ;
                        ;  Finished
                        ;
                        9$:     JMP     CLQXIT          ;Finished with .SPFUN
                        ;
                        ;-------------------------------------------------------------------------
                        ;  Special function # 203
                        ;  Read with Q.WCNT indicating the byte count rather than the word count
                        ;
                        SFREAD: BIC     #CM$EOF,CL$STA(R5);Clear end of file status
                                JMP     CLREAD          ;Enter read routine (Q.WCNT = byte count)
                        ;
                        ;-------------------------------------------------------------------------
                        ;  Special function # 204
                        ;  Get handler status.
                        ;  The following information is stored into the first word of the
                        ;  user's buffer:
                        ;   High order byte: Handler version number
                        ;   Low order byte:
                        ; XL$XFX bit 0:  1 ==> We have sent XOFF to stop transmission to us.
                        ; XL$XFR bit 1:  1 ==> We have received an XOFF.
                        ; XL$CTS bit 2:  1 ==> Clear To Send (CTS) is asserted.
                        ;  Next two are RT 5.4 compatible
                        ; XL$CD  bit 3:  1 ==> Carrier is detected
                        ; XL$RI  bit 4:  1 ==> Ring is detected
                        ;
                        SFSTAT: MOV     R2,-(SP)
                        ;
                                CALL    CLGSTS          ;Call common routine to get status
                                CALL    CLPTWD          ;Store value into user's buffer
                        ;
                        ;  Finished
                        ;
                                MOV     (SP)+,R2
                                JMP     CLQXIT          ;Finished with operation
                        ;
                        ;-------------------------------------------------------------------------
                        ;  Special function # 205
                        ;  Terminate I/O to the line.
                        ;
                        SFTERM:
                        ;
                        ;  Set flag saying to ignore input from the line
                        ;
                                BIC     #CM$ON,CL$STA(R5);Say line is turned off
                        ;
                        ;  Clear input and output silos and other CL unit status
                        ;
                                CALL    CLREST          ;Reset the CL unit
                        ;
                        ;  Drop Data Terminal Ready
                        ;
                                BIC     #CO$DTR,CL$OPT(R5) ;Say we want DTR off
```

```
                                        CALL    SETDTR          ;Call routine to drop DTR
                                ;
                                ;  Finished
                                ;
                                        JMP     CLQXIT
                                ;-------------------------------------------------------------------------
                                ;  Special function # 206
                                ;  Raise or drop DTR signal.
                                ;
                                SFDTR:  TST     Q.WCNT-Q.BLKN(R4) ;Raise or drop DTR?
                                        BNE     1$              ;Br if raising DTR
                                ;
                                ;  Drop DTR
                                ;
                                        BIC     #CO$DTR,CL$OPT(R5) ;Drop DTR
                                        BR      2$
                                ;
                                ;  Raise DTR
                                ;
                                1$:     BIS     #CO$DTR,CL$OPT(R5) ;Raise DTR
                                2$:     CALL    SETDTR          ;Call routine to raise or drop DTR
                                        JMP     CLQXIT          ;Finished with .SPFUN
                                ;-------------------------------------------------------------------------
                                ;  Special function # 250
                                ;  Set option flags
                                ;
                                SFSOPT: CALL    GETWRD          ;Get word from user's buffer
                                        BIS     R0,CL$OPT(R5)   ;Set specified option flags
                                        CALL    SETDTR          ;Check for DTR status change
                                        JMP     CLQXIT
                                ;
                                ;-------------------------------------------------------------------------
                                ;  Special function # 251
                                ;  Clear option flags
                                ;
                                SFCOPT: CALL    GETWRD          ;Get word from user's buffer
                                        BIC     R0,CL$OPT(R5)   ;Clear specified option flags
                                        CALL    SETDTR          ;Check for DTR status change
                                        JMP     CLQXIT
                                ;
                                ;-------------------------------------------------------------------------
                                ;  Special function # 252
                                ;  Set page length
                                ;
                                SFSLEN: CALL    GETWRD          ;Get word from user's buffer
                                        MOV     R0,CL$LEN(R5)   ;Set page length for this unit
                                        JMP     CLQXIT
                                ;
                                ;-------------------------------------------------------------------------
                                ;  Special function # 253
                                ;  Set number of lines to skip at bottom of page.
                                ;
                                SFSSKP: CALL    GETWRD          ;Get word from user's buffer
                                        MOV     R0,CL$SKP(R5)   ;Set skip lines
                                        JMP     CLQXIT
```

```
                                        ;
                                        ;-------------------------------------------------------------------------
                                        ;  Special function # 254
                                        ;  Set line width.
                                        ;
000760  004737  001634'       SFSWID: CALL    GETWRD          ;Get word from user's buffer
000764  010065  0000000              MOV     R0,CL$WID(R5)   ;Set line width
000770  000137  001534'              JMP     CLQXIT
                                        ;
                                        ;-------------------------------------------------------------------------
                                        ;  Special function # 255
                                        ;  Get modem status
                                        ;
000774  010246                  SFGMS:  MOV     R2,-(SP)
                                        ;
                                        ;  Call hardware dependent routine to get the modem status
                                        ;
000776  004737  0000000              CALL    GETDSS          ;Call routine to get the data set status
                                        ;
                                        ;  Return status value to 1st word of user's buffer
                                        ;
001002  004737  001572'              CALL    CLPTWD          ;Store value into 1st word of user's buffer
                                        ;
                                        ;  Finished
                                        ;
001006  012602                       MOV     (SP)+,R2
001010  000137  001534'              JMP     CLQXIT          ;Finished I/O operation
                                        ;
                                        ;-------------------------------------------------------------------------
                                        ;  Special function # 256.
                                        ;  Set transmit/receive speed.
                                        ;
001014  004737  001634'       SFSPD:  CALL    GETWRD          ;Get word from user's buffer
001020  103402                       BCS     1$              ;Br if invalid buffer address
001022  004737  0000000              CALL    SETSPD          ;Set the speed
001026  000137  001534'       1$:     JMP     CLQXIT          ;Finished
                                        ;
                                        ;-------------------------------------------------------------------------
                                        ;  Special function # 257.
                                        ;  Abort all pending read and write requests for the job.
                                        ;
                                        ;  Inputs:
                                        ;   R4 = Pointer to 3rd word of .SPFUN queue element.
                                        ;
001032  010446                  SFABT:  MOV     R4,-(SP)        ;Save pointer to current queue element
001034  116404  000000C              MOVB    Q.JOB-Q.BLKW(R4),R4 ;Get job # from .SPFUN queue element
                                        ;
                                        ;  Abort pending read requests for this job
                                        ;
001040  012703  0000000              MOV     #CL$RQH,R3      ;Point to read queue head
001044  004737  005614'              CALL    CKABTQ          ;Abort pending reads for job
                                        ;
                                        ;  Abort pending write requests for this job
                                        ;
001050  012703  0000000              MOV     #CL$WQH,R3      ;Point to write queue head
001054  004737  005614'              CALL    CKABTQ          ;Abort pending writs for job
                                        ;
```

```
                                        ;  Call routine to return any freed queue elements to the system
                                        ;
001060  004737  006002'                         CALL    RTNQ            ;Return freed queue elements to the system
                                        ;
                                        ;  Finished
                                        ;
001064  012604                          10$:    MOV     (SP)+,R4        ;Restore pointer to queue element
001066  000137  001534'                         JMP     CLQXIT          ;Finished operation
                                        ;
                                        ;-------------------------------------------------------------------------
                                        ;  Special function # 261.
                                        ;  Get number of bytes pending in input silo buffer.
                                        ;
001072  016100  000000G         SFIC:   MOV     LHIRBA(R1),R0   ;Get allocated size of input buffer
001076  166100  000000G                 SUB     LHIRBS(R1),R0   ;Subtract free space to get # chars in buf
001102  004737  001572'                 CALL    CLPTWD          ;Store value into user's buffer
001106  000137  001534'                 JMP     CLQXIT          ;Finished with operation
                                        ;
                                        ;-------------------------------------------------------------------------
                                        ;  Special function # 262.
                                        ;  Get number of bytes pending in output ring buffer.
                                        ;
001112  010446                  SFOC:   MOV     R4,-(SP)
001114  016500  000000G                 MOV     CL$ORA(R5),R0   ;Get allocated space for output ring buffer
001120  166500  000000G                 SUB     CL$ORS(R5),R0   ;Subtract free space to get # chars in buf
001124  016504  000000G                 MOV     CL$LIX(R5),R4   ;Get index # of line we are assigned to
001130  001412                          BEQ     1$              ;Br if not assigned to a line
001132  032765  000000G 000000G        BIT     #CM$EFP,CL$STA(R5);Are we doing end-of-file processing?
001140  001401                          BEQ     2$              ;Br if not
001142  005200                          INC     R0              ;Add an extra character
001144  032764  000000G 000000G 2$:     BIT     #$XCHAR,LSW3(R4);Is output transmission going to line now?
001152  001401                          BEQ     1$              ;Br if not
001154  005200                          INC     R0              ;Say another character pending for output
001156  012604                  1$:     MOV     (SP)+,R4        ;Restore pointer into queue element
001160  004737  001572'                 CALL    CLPTWD          ;Store value into user's buffer
001164  000137  001534'                 JMP     CLQXIT          ;Finished with operation
                                ;
                                ;-------------------------------------------------------------------------
                                ;  Special function # 264.
                                ;  Set end-of-file output processing control information.
                                ;
001170                          SFSEFP:
                                ;
                                ;  Set form-feed count
                                ;
001170  004737  001634'                 CALL    GETWRD          ;Get form-feed count from user's buffer
001174  103422                          BCS     9$              ;Br if invalid buffer address
001176  120027  000377                  CMPB    R0,#377         ;Don't change form-feed count?
001202  001402                          BEQ     2$              ;Br if don't-change value
001204  010065  000000G                 MOV     R0,CL$EPN(R5)   ;Set # form-feeds to send at end-of-file
                                ;
                                ;  Set up end-of-file output string
                                ;
001210  016502  000000G         2$:     MOV     CL$EPS(R5),R2   ;Get pointer to area where string is stored
001214  012703  000000G                 MOV     #CLEOFS,R3      ;Get max # bytes allowed for string
001220  004737  000000G         1$:     CALL    GTBYT           ;Get next byte from string
001224  121627  000377                  CMPB    (SP),#377       ;Don't change string?
```

```
 286 001230  001404                        BEQ     9$              ;Br if don't change
 287 001232  112622                        MOVB    (SP)+,(R2)+     ;Move char to string area
 288 001234  001402                        BEQ     9$              ;Br if this is end of string
 289 001236  077310                        SOB     R3,1$           ;Loop if we can get more chars
 290 001240  105022                        CLRB    (R2)+           ;Terminate string with null
 291                                ;
 292                                ;  Finished
 293                                ;
 294 001242  000137  001534'        9$:     JMP     CLQXIT          ;Finished
 295                                ;
 296                                ;-------------------------------------------------------------------------
 297                                ;  Special function # 265.
 298                                ;  Reset CL unit.
 299                                ;
 300 001246  004737  001756'        SFREST: CALL    CLREST          ;Call routine to reset CL unit status
 301 001252  000137  001534'                JMP     CLQXIT          ;Finished
 302                                ;
 303                                ;-------------------------------------------------------------------------
 304                                ;  Special function # 266
 305                                ;  Get current options and settings.
 306                                ;
 307                                ;  Returns 13 words to user buffer
 308                                ;  1    Handler status as for SPFUN 204
 309                                ;  2    CL options flags
 310                                ;  3    internal flags word
 311                                ;  4    page length
 312                                ;  5    end of page skip lines
 313                                ;  6    page width
 314                                ;  7    TS line number and CL unit number
 315                                ;  8    number end of file form feeds
 316                                ;  9-12 end of file string, ASCIZ up to CLEOFS long
 317                                ;
 318 001256                         SFGOPT:
 319                                ;
 320                                ;  Get and return status word just as for SPFUN 204
 321                                ;  word 1
 322 001256  004737  001432'                CALL    CLGSTS          ;Get current status word
 323 001262  004737  001572'                CALL    CLPTWD          ;Return to user buffer
 324 001266  103457                         BCS     9$              ;Error return if no chan or odd buff addr
 325                                ;
 326                                ;  Return CL options word
 327                                ;  word 2
 328 001270  016500  0000000                MOV     CL$OPT(R5),R0   ;Get options word
 329 001274  004737  001572'                CALL    CLPTWD          ;Return to user buffer
 330                                ;
 331                                ;  Return internal status word
 332                                ;  word 3
 333 001300  016500  0000000                MOV     CL$STA(R5),R0   ;Get internal status word
 334 001304  004737  001572'                CALL    CLPTWD          ;Return to user buffer
 335                                ;
 336                                ;  Return current page length
 337                                ;  word 4
 338 001310  016500  0000000                MOV     CL$LEN(R5),R0   ;Get current length
 339 001314  004737  001572'                CALL    CLPTWD          ;Return to user buffer
 340                                ;
 341                                ;  Return current number lines to skip at end of page
 342                                ;  word 5
```

```
343 001320  016500  0000000                 MOV     CL$SKP(R5),R0   ;Get current # skip lines
344 001324  004737  001572'                 CALL    CLPTWD          ;Return to user buffer
345                                 ;
346                                 ;  Return current page width
347                                 ;  word 6
348 001330  016500  0000000                 MOV     CL$WID(R5),R0   ;Get current width
349 001334  004737  001572'                 CALL    CLPTWD          ;Return to user buffer
350                                 ;
351                                 ;  Line number being used as CL unit --> low byte
352                                 ;  CL unit number --> high byte (will be > 7 if C1 unit);
353                                 ;  word 7
354 001340  010146                          MOV     R1,-(SP)        ;Get T/S line index
355 001342  010500                          MOV     R5,R0           ;Get CL unit index
356 001344  000300                          SWAB    R0              ;Move to high byte
357 001346  052600                          BIS     (SP)+,R0        ;Merge in line index
358 001350  006200                          ASR     R0              ;Convert indices to numbers
359 001352  004737  001572'                 CALL    CLPTWD          ;Return to user buffer
360                                 ;
361                                 ;  Return current # end of file form feeds
362                                 ;  word 8
363 001356  016500  0000000                 MOV     CL$EPN(R5),R0   ;Get EOF FF's
364 001362  004737  001572'                 CALL    CLPTWD          ;Return to user buffer
365                                 ;
366                                 ;  Return current end of file string
367                                 ;  words 9 - 12
368 001366  010246                          MOV     R2,-(SP)        ;Save registers
369 001370  010346                          MOV     R3,-(SP)
370 001372  012703  0000000                 MOV     #CLEOFS,R3      ;Get number of chars to move
371 001376  016502  0000000                 MOV     CL$EPS(R5),R2   ;Get pointer to EOF string
372 001402  112246          1$:            MOVB    (R2)+,-(SP)     ;Get next character
373 001404  001404                          BEQ     2$              ;Stop at end of string
374 001406  004737  0000000                 CALL    PTBYT           ;Else move to user buffer
375 001412  077305                          SOB     R3,1$           ;Move up to maximum length
376 001414  005046                          CLR     -(SP)           ;Always return ASCIZ string
377 001416  004737  0000000         2$:     CALL    PTBYT           ;Move last char to user buffer
378 001422  012603                          MOV     (SP)+,R3        ;Restore registers
379 001424  012602                          MOV     (SP)+,R2
380                                 ;
381                                 ;  Finished
382                                 ;
383 001426  000137  001534'         9$:     JMP     CLQXIT
```

```
                                        .SBTTL  CLGSTS -- Return CL device status
                                ;-----------------------------------------------------------------------
                                ;  CLGSTS is called by CL .SPFUNs 204 and 266 to return the CL version
                                ;  number and modem status bits in R0.
                                ;  Inputs:
                                ;       R1 index number of line being used as CL unit
                                ;       R5 contains the CL unit index number
                                ;  Outputs:
                                ;       R0 contains the version and status bits
                                ;          (see .SPFUN 204 for complete bit description)
                                ;
001432  010346                  CLGSTS: MOV     R3,-(SP)        ;Save R3
                                ;
                                ;  Get version number to high-order byte
                                ;
001434  113703  000000G                 MOVB    CLVERS,R3       ;Get version number
001440  042703  177400                  BIC     #^C377,R3       ;Kill possible sign extension
001444  000303                          SWAB    R3              ;Move version to high byte
                                ;
                                ;  See if we have sent an XOFF to stop transmission to us
                                ;
001446  032761  000000G 000000G         BIT     #$HISTP,LSWIO(R1);Have we send XOFF?
001454  001402                          BEQ     1$              ;Br if not
001456  052703  000000G                 BIS     #XL$XFX,R3      ;Set status flag
                                ;
                                ;  See if we have received an XOFF
                                ;
001462  032761  000000G 000000G 1$:     BIT     #$CTRLS,LSW3(R1);Have we received an XOFF?
001470  001402                          BEQ     2$              ;Br if not
001472  052703  000000G                 BIS     #XL$XFR,R3      ;Set status flag
                                ;
                                ;  See if Clear To Send (CTS) is asserted
                                ;
001476  004737  000000G         2$:     CALL    GETDSS          ;Call routine to get dataset status
001502  032700  000000G                 BIT     #MS$CAR,R0      ;Is carrier detected?
001506  001402                          BEQ     3$              ;Br if not
001510  052703  000000C                 BIS     #<XL$CTS!XL$CD>,R3 ;Say CTS is asserted and ring detected
                                ;
                                ;  See if Ring is asserted
                                ;
001514  032700  000000G         3$:     BIT     #MS$RNG,R0      ;Is ring detected?
001520  001402                          BEQ     4$              ;Br if not
001522  052703  000000G                 BIS     #XL$RI,R3       ;Say ring is detected
                                ;
                                ;  Return status value in R0
                                ;
001526  010300                  4$:     MOV     R3,R0           ;Get value to R0 for CLPTWD
001530  012603                          MOV     (SP)+,R3        ;Restore R3
001532  000207                          RETURN
```

```
                                ;-------------------------------------------------------------------------------
                                ;  We completed the I/O operation.
                                ;  Return the queue element to the system.
                                ;
                                ;  Inputs:
                                ;   R4 = Address of current queue element.
                                ;
001534                          CLQXIT: DISABL                      ;;;** Disable interrupts **
001542  013764  000004' 000000C         MOV     CQH,Q.LINK-Q.BLKN(R4);;;Put queue element on completed list
001550  010437  000004'                 MOV     R4,CQH              ;;;
001554                                  ENABL                       ;** Enable interrupts
001562  004737  006002'                 CALL    RTNQ                ;Return queue element to the system
                                ;
                                ;  Go back and see if there is another queue element pending
                                ;
001566  000137  000016'                 JMP     CLQOK               ;Go back and check for another request
```

```
                                        .SBTTL  CLPTWD -- Store 1 word into user's buffer
                                ;-----------------------------------------------------------------------
                                ;  CLPTWD is called from some of the .SPFUN processing routines to store
                                ;  a one word value into the 1st word of the user's data buffer.
                                ;  If the buffer address is odd, the error flag is set in the channel
                                ;  status word, the C-flag is set on return, and the value is not stored.
                                ;
                                ;  Inputs:
                                ;   R0 = Value to store.
                                ;   R4 = Pointer to current queue element.
                                ;
                                ;  Outputs:
                                ;   C-flag set ==> Error:  buffer address odd
                                ;
001572                          CLPTWD:
                                ;
                                ;  See if the buffer address is odd
                                ;
001572  032764  000001  000000C        BIT     #1,Q.BUFF-Q.BLKN(R4) ;Is the buffer address odd?
001600  001410                         BEQ     1$                  ;Br if not
                                ;
                                ;  Error:  The buffer address is odd
                                ;
001602  016400  000000C                MOV     Q.CSW-Q.BLKN(R4),R0 ;Get address of CSW for channel
001606  001403                         BEQ     2$                  ;Br if no channel address
001610  052760  0000000 0000000G      BIS     #CS$ERR,C.CSW(R0);Set error flag in CSW
001616  000261                  2$:    SEC                         ;Signal error on return
001620  000404                         BR      9$
                                ;
                                ;  Buffer address is OK.
                                ;  Call PTWRD to store the value.
                                ;
001622  010046                 1$:    MOV     R0,-(SP)            ;Stack the value for PTWRD
001624  004737  0000000G              CALL    PTWRD               ;Store value into user's buffer
001630  000241                         CLC                         ;Signal success on return
                                ;
                                ;  Finished
                                ;
001632  000207                 9$:    RETURN
```

```
                                        .SBTTL  GETWRD -- Get 1 word from user's buffer
                                ;-----------------------------------------------------------------------
                                ;  GETWRD is called from some of the .SPFUN processing routines to get
                                ;  a one word value from the 1st word of the user's data buffer.
                                ;  If the buffer address is odd, the error flag is set in the channel
                                ;  status word, the C-flag is set on return, and 0 (zero) is returned
                                ;  in R0.
                                ;
                                ;  Inputs:
                                ;   R4 = Pointer to current queue element
                                ;
                                ;  Outputs:
                                ;   R0 = Value from 1st word of data buffer
                                ;   C-flag set ==> Buffer address was odd (R0 contains 0 in this case).
                                ;   Buffer address is incremented by 2 in queue element.
                                ;
001634                          GETWRD:
                                ;
                                ;  See if the buffer address is odd
                                ;
001634  032764  000001  000000C         BIT     #1,Q.BUFF-Q.BLKN(R4)    ;Is the buffer address odd?
001642  001411                          BEQ     1$                      ;Br if not odd
                                ;
                                ;  Error: The buffer address is odd
                                ;
001644  016400  000000C                 MOV     Q.CSW-Q.BLKN(R4),R0     ;Get address of channel status word
001650  001403                          BEQ     2$                      ;Br if there is none
001652  052760  000000G 000000G         BIS     #CS$ERR,C.CSW(R0)       ;Set error flag in channel status
001660  005000                  2$:     CLR     R0                      ;Return 0 in R0
001662  000261                          SEC                             ;Signal error on return
001664  000411                          BR      9$
                                ;
                                ;  Buffer address is ok.
                                ;  Map PAR6 to user's buffer.
                                ;
001666  016437  000000C 000000G 1$:     MOV     Q.PAR-Q.BLKN(R4),@#KPAR6 ;Map KPAR6 to user's buffer
                                ;
                                ;  Get word from the buffer
                                ;
001674  017400  000000C                 MOV     @Q.BUFF-Q.BLKN(R4),R0   ;Get value from buffer
001700  062764  000002  000000C         ADD     #2,Q.BUFF-Q.BLKN(R4)    ;Advance buffer address
001706  000241                          CLC                             ;Signal success on return
                                ;
                                ;  Finished
                                ;
001710  000207                  9$:     RETURN
```

```
                                        .SBTTL  CLCLOS -- Initiate end-of-file processing
                                ;-----------------------------------------------------------------------
                                ;  CLCLOS is called when end of file is reached on output processing
                                ;  and we want to initiate the end-of-file output processing.
                                ;
                                ;  Inputs:
                                ;   R5 = CL unit index
                                ;
001712                          CLCLOS:
                                ;
                                ;  Only do output EOF processing if a write was done to this unit
                                ;
001712  032765  0000000 0000000         BIT     #CM$WRT,CL$STA(R5);Was a write done to this unit?
001720  001415                         BEQ     9$                ;Br if not
                                ;
                                ;  Say we are doing end-of-file processing
                                ;
001722  052765  0000000 0000000         BIS     #CM$EFP,CL$STA(R5);We have started EOF processing for unit
001730  042765  0000000 0000000         BIC     #CM$WRT,CL$STA(R5);Clear write-done flag for unit
                                ;
                                ;  Reset form-feed count
                                ;
001736  105065  0000010                 CLRB    CL$EPN+1(R5)    ;Say no form-feeds sent yet
                                ;
                                ;  Reset ENDSTRING pointer
                                ;
001742  016565  0000000 0000000         MOV     CL$EPS(R5),CL$EPP(R5);Reset endstring pointer
                                ;
                                ;  Initiate output to the unit
                                ;
001750  004737  003122'                 CALL    ORINGP          ;Initiate output to unit
                                ;
                                ;  Finished
                                ;
001754  000207                  9$:     RETURN
```

```
                                        .SBTTL  CLREST -- Reset a CL unit
                                ;-----------------------------------------------------------------------
                                ;  Reset a CL unit.  This consists of the following actions:
                                ;   1. Empty input silo.
                                ;   2. Empty output silo.
                                ;   3. Reset line and column positions.
                                ;   4. Stop sending break if we are currently sending it.
                                ;   5. Clear flag that says we have received an XOFF.
                                ;   6. Send an XON if we previously sent an XOFF.
                                ;
                                ;  Inputs:
                                ;   R5 = CL unit number index
                                ;
001756  010146                  CLREST: MOV     R1,-(SP)
                                ;
                                ;  Get line # CL unit is connected to
                                ;
001760  016501  000000G                 MOV     CL$LIX(R5),R1   ;Get line index number
                                ;
                                ;  Clear out the input silo buffer
                                ;
001764                                  DISABL                  ;;;** Disable interrupts **
001772  016100  000000G                 MOV     LHIRBB(R1),R0   ;;;Get pointer to start of silo buffer
001776  010061  000000G                 MOV     R0,LHIRBP(R1)   ;;;Reset input interrupt pointer
002002  010061  000000G                 MOV     R0,LHIRBG(R1)   ;;;Reset next available char pointer
002006  016161  000000G 000000G         MOV     LHIRBA(R1),LHIRBS(R1);;;Reset free space counter
                                ;
                                ;  Clear out the output silo buffer
                                ;
002014  016500  000000G                 MOV     CL$ORB(R5),R0   ;;;Get pointer to start of output silo buffer
002020  010065  000000G                 MOV     R0,CL$ORP(R5)   ;;;Output character pointer
002024  010065  000000G                 MOV     R0,CL$ORG(R5)   ;;;Next available output character
002030  016565  000000G 000000G         MOV     CL$ORA(R5),CL$ORS(R5);;;Available space in output buffer
002036                                  ENABL                   ;** Enable interrupts **
                                ;
                                ;  Clear flag that says we have received an XOFF
                                ;
002044  042761  000000G 000000G         BIC     #^C<CTRLS>,LSW3(R1);Say line output not suspended due to XOFF
                                ;
                                ;  Clear some status flags for the unit
                                ;
002052  042765  000000C 000000G         BIC     #<CM$WRT!CM$EFP!CM$CRL!CM$TBS!CM$EOF!CM$FFS>,CL$STA(R5)
                                ;
                                ;  If we are sending a break, stop now
                                ;
002060  032765  000000G 000000G         BIT     #CM$BRK,CL$STA(R5);Are we sending a break now?
002066  001406                          BEQ     1$              ;Br if not
002070  005000                          CLR     R0              ;Say to reset break transmission
002072  004737  006340'                 CALL    SETBRK          ;Stop sending break
002076  042765  000000G 000000G         BIC     #CM$BRK,CL$STA(R5);Say break transmission finished
                                ;
                                ;  Reset page and line position
                                ;
002104  005065  000000G         1$:     CLR     CL$LIN(R5)      ;Say we are at the top of a page
002110  005065  000000G                 CLR     CL$COL(R5)      ;Say we are at left-most column of line
                                ;
                                ;  If we previously sent an XOFF to stop the sender, send an XON now.
```

```
     58                                         ;
     59 002114  032761  0000000 0000000        BIT     #$HISTP,LSWIO(R1);Did we send an XOFF?
     60 002122  001407                          BEQ     9$                ;Br if not
     61 002124  042761  0000000 0000000        BIC     #$HISTP,LSWIO(R1);Can XOFF has been cleared
     62 002132  016100  0000000                 MOV     LCDTYP(R1),R0     ;Get line type index
     63 002136  004770  0000000                 CALL    @CDSXON(R0)       ;Send XON
     64                                    ;
     65                                    ;  Finished
     66                                    ;
     67 002142  012601                     9$:   MOV     (SP)+,R1
     68 002144  000207                           RETURN
```

```
                                        .SBTTL  CLINCP -- Input character processing
                                ;-----------------------------------------------------------------
                                ;  CLINCP is called at fork level after each received character has been
                                ;  stored in the input silo buffer.  Its primary function is to move
                                ;  characters from the input silo buffer to the user's data buffer.
                                ;
                                ;  Inputs:
                                ;   R4 = Line index number of line that received a character.
                                ;
002146  010146                  CLINCP: MOV     R1,-(SP)
002150  010546                          MOV     R5,-(SP)
                                ;
                                ;  Convert line index number to CL unit index
                                ;
002152  016405  0000000                 MOV     LCLUNT(R4),R5   ;Carry CL unit number in R5
                                ;
                                ;  If this CL unit is cross connected to a time-sharing line, try to
                                ;  start output to the time-sharing line (it will fetch characters
                                ;  directly from the input silo for the CL unit).
                                ;
002156  016501  0000000                 MOV     CL$XLN(R5),R1   ;Is this CL unit cross-connected to TT line?
002162  001403                          BEQ     1$              ;Br if not
002164  004777  0000000                 CALL    @TRNSTR         ;Try to start output to TT line
002170  000402                          BR      9$
                                ;
                                ;  See if we need to move any characters from the input silo buffer
                                ;  to the user's data buffer
                                ;
002172  004737  002204'         1$:     CALL    IRINGG          ;Move chars to user's data buffer
                                ;
                                ;  Finished
                                ;
002176  012605                  9$:     MOV     (SP)+,R5
002200  012601                          MOV     (SP)+,R1
002202  000207                          RETURN
```

```
                                        .SBTTL  IRINGG -- Move chars from silo buffer to data buffer
                                ;-------------------------------------------------------------------
                                ;  IRINGG is called to move all characters from the terminal input
                                ;  silo buffer to the current read data buffer.
                                ;
                                ;  Inputs:
                                ;   R5 = CL unit index number
                                ;
002204                          IRINGG:
                                ;
                                ;  See if this routine is already being used by this unit.
                                ;  If so, don't reenter it (the other process will transfer all characters).
                                ;
002204                                  DISABL                  ;;;** Disable interrupts **
002212  032765  0000000 0000000         BIT     #CM$IRG,CL$STA(R5) ;;;Is this routine already active for unit?
002220  001404                          BEQ     2$              ;;;Br if not
002222                                  ENABL                   ;** Enable interrupts **
002230  000207                          RETURN
                                ;
                                ;  This routine is not active, claim it for us
                                ;
002232  052765  0000000 0000000 2$:     BIS     #CM$IRG,CL$STA(R5) ;;;Say the routine is now active
002240                                  ENABL                   ;** Enable interrupts **
                                ;
                                ;  Push some registers
                                ;
002246  010146                          MOV     R1,-(SP)
002250  010246                          MOV     R2,-(SP)
002252  010346                          MOV     R3,-(SP)
002254  010446                          MOV     R4,-(SP)
                                ;
                                ;  Get index number of line associated with this CL unit
                                ;
002256  016501  0000000                 MOV     CL$LIX(R5),R1   ;Get line index number
                                ;
                                ;  See if there are any characters in the input buffer and if there
                                ;  is a pending read request for this unit.
                                ;
002262                          3$:     DISABL                  ;;;** Disable interrupts **
002270  016504  0000000                 MOV     CL$RQH(R5),R4   ;;;Is there a pending read request?
002274  001475                          BEQ     9$              ;;;Br if not
002276  032765  0000000 0000000         BIT     #CM$EOF,CL$STA(R5);;;Need to report end of file?
002304  001004                          BNE     7$              ;;;Br if yes
002306  026161  0000000 0000000         CMP     LHIRBS(R1),LHIRBA(R1);;;Any chars in the silo buffer?
002314  001465                          BEQ     9$              ;;;Br if not
                                ;
                                ;  There are characters in the silo buffer and there is a pending
                                ;  read request.
                                ;
002316                          7$:     ENABL                   ;** Enable interrupts **
                                ;
                                ;  See if flag is set which indicates that we should signal end-of-file
                                ;
002324  032765  0000000 0000000         BIT     #CM$EOF,CL$STA(R5);Should we signal end of file?
002332  001413                          BEQ     4$              ;Br if not
002334  016403  000000C                 MOV     Q.CSW-Q.BLKN(R4),R3;Get pointer to CSW for channel
002340  052763  0000000 0000000         BIS     #CS$EOF,C.CSW(R3);Set end of file flag
```

```
     58 002346  042765  000000G 000000G          BIC     #CM$EOF,CL$STA(R5);Acknowledge the EOF
     59 002354  004737  002734'                 CALL    RDFIN           ;Terminate this read operation
     60 002360  000740                          BR      3$              ;See if there is another read to do
     61                                 ;
     62                                 ;  Get a character from the silo buffer
     63                                 ;
     64 002362  004777  000000G         4$:     CALL    @SILFET         ;Get a character from input silo
     65 002366  103440                          BCS     9$              ;Br if no chars in silo
     66 002370  010002                          MOV     R0,R2           ;Get character to R2
     67                                 ;
     68                                 ;  If this is a control character, do special processing
     69                                 ;
     70 002372  020227  000032          6$:     CMP     R2,#32          ;Is this a control character?
     71 002376  101017                          BHI     5$              ;Br if not
     72 002400  105702                          TSTB    R2              ;Is this a null character?
     73 002402  001004                          BNE     8$              ;Br if not null
     74 002404  032765  000000G 000000G        BIT     #CO$BNI,CL$OPT(R5);Is binary input wanted?
     75 002412  001723                          BEQ     3$              ;Br if not -- ignore nulls
     76 002414  126427  000000C 000000G 8$:     CMPB    Q.FUNC-Q.BLKN(R4),#CLSFRB ;Is this a special read (.SPFUN 203)
     77 002422  001405                          BEQ     5$              ;If yes then accept control chars as normal
     78 002424  010200                          MOV     R2,R0           ;Get the control character
     79 002426  006300                          ASL     R0              ;Convert to word table index
     80 002430  004770  002516'                 CALL    @CCIRTN(R0)     ;Call control character processing routine
     81 002434  000712                          BR      3$              ;Go see if there are more characters
     82                                 ;
     83                                 ;  This is not a control character
     84                                 ;  Store into user's data buffer.
     85                                 ;
     86 002436  004737  002700'         5$:     CALL    INPCHR          ;Store character into data buffer
     87                                 ;
     88                                 ;  If the input silo buffer is now empty, and this is a special function
     89                                 ;  read (.SPFUN 203), then say the read is finished.
     90                                 ;
     91 002442  126427  000000C 000000G        CMPB    Q.FUNC-Q.BLKN(R4),#CLSFRB ;is this a special read (.SPFUN 203)
     92 002450  001304                          BNE     3$              ;Br if not -- continue reading more
     93 002452  026161  000000G 000000G        CMP     LHIRBS(R1),LHIRBA(R1);Is the silo buffer empty?
     94 002460  001300                          BNE     3$              ;Br if not -- Get more chars for the SPFUN
     95 002462  004737  002734'                 CALL    RDFIN           ;Terminate the read operation
     96 002466  000675                          BR      3$              ;See if there is another read request
     97                                 ;
     98                                 ;  There are no more input characters that can be moved from silo buffer.
     99                                 ;  Say this routine is no longer active for this unit.
    100                                 ;
    101 002470  042765  000000G 000000G 9$:     BIC     #CM$IRG,CL$STA(R5) ;;;Say we are leaving this routine
    102 002476                                  ENABL                   ;** Enable interrupts **
    103                                 ;
    104                                 ;  Finished
    105                                 ;
    106 002504  012604                          MOV     (SP)+,R4
    107 002506  012603                          MOV     (SP)+,R3
    108 002510  012602                          MOV     (SP)+,R2
    109 002512  012601                          MOV     (SP)+,R1
    110 002514  000207                          RETURN
```

```
                                        .SBTTL  CCIRTN -- Input control character processing routines
                                ;-----------------------------------------------------------------------
                                ;  These routines are called to process control characters received
                                ;  from a line.
                                ;
                                ;  Inputs:
                                ;   R2 = Control character
                                ;   R5 = Unit index number
                                ;
                                ;  Vector of control character processing routines
                                ;
002516  002612'         CCIRTN: .WORD   CCINUL           ;00 null
002520  002604'                 .WORD   CCISTR           ;01 SHO
002522  002604'                 .WORD   CCISTR           ;02 STX
002524  002604'                 .WORD   CCISTR           ;03 ETX
002526  002604'                 .WORD   CCISTR           ;04 EOT
002530  002604'                 .WORD   CCISTR           ;05 ENQ
002532  002604'                 .WORD   CCISTR           ;06 ACK
002534  002604'                 .WORD   CCISTR           ;07 BEL
002536  002604'                 .WORD   CCISTR           ;10 BACKSPACE
002540  002604'                 .WORD   CCISTR           ;11 TAB
002542  002624'                 .WORD   CCILF            ;12 LINE FEED
002544  002604'                 .WORD   CCISTR           ;13 VT
002546  002604'                 .WORD   CCISTR           ;14 FF
002550  002636'                 .WORD   CCICR            ;15 CARRIAGE RETURN
002552  002604'                 .WORD   CCISTR           ;16 SO
002554  002604'                 .WORD   CCISTR           ;17 SI
002556  002604'                 .WORD   CCISTR           ;20 DLE
002560  002604'                 .WORD   CCISTR           ;21 XON
002562  002604'                 .WORD   CCISTR           ;22 DC2
002564  002604'                 .WORD   CCISTR           ;23 XOFF
002566  002604'                 .WORD   CCISTR           ;24 DC4
002570  002604'                 .WORD   CCISTR           ;25 NAK
002572  002604'                 .WORD   CCISTR           ;26 SYN
002574  002604'                 .WORD   CCISTR           ;27 ETB
002576  002604'                 .WORD   CCISTR           ;30 CAN
002600  002604'                 .WORD   CCISTR           ;31 EM
002602  002664'                 .WORD   CCICTZ           ;32 SUB (ctrl-Z)
```

```
                                        ;
                                        ;  Routine to store the control character
                                        ;
002604  004737  002700'         CCISTR: CALL    INPCHR          ;Store the character
002610  000207                          RETURN
                                        ;
                                        ;  Routine to process a null character
                                        ;
002612  032765  000000G 000000G CCINUL: BIT     #CO$BNI,CL$OPT(R5);Are we in binary input mode?
002620  001371                          BNE     CCISTR          ;Br if yes -- go store the null
002622  000207                          RETURN                  ;Discard the null
                                        ;
                                        ;  Routine to process a line feed
                                        ;
002624  032765  000000G 000000G CCILF:  BIT     #CO$LFI,CL$OPT(R5);Should we ignore input line feeds?
002632  001364                          BNE     CCISTR          ;Br if not
002634  000207                          RETURN                  ;Discard the LF
                                        ;
                                        ;  Routine to process carriage returns
                                        ;
002636  016500  000000G         CCICR:  MOV     CL$RQH(R5),R0   ;Get address of current Q element
002642  126027  000000C 000000G        CMPB    Q.FUNC-Q.BLKN(R0),#CLSFRL ;Read-line special function?
002650  001355                          BNE     CCISTR          ;Br if not -- Treat CR as normal char
002652  004737  002700'                 CALL    INPCHR          ;Store the carriage return
002656  004737  002734'                 CALL    RDFIN           ;Terminate the read operation
002662  000207                          RETURN
                                        ;
                                        ;  Routine to process control-Z characters
                                        ;
002664                          CCICTZ:
                                        ;
                                        ;  Set flag which will cause us to return EOF status on next read
                                        ;
002664  052765  000000G 000000G        BIS     #CM$EOF,CL$STA(R5);Remember EOF has been hit
                                        ;
                                        ;  Terminate this read operation
                                        ;
002672  004737  002734'                 CALL    RDFIN           ;Terminate the read operation
002676  000207                          RETURN
```

```
                                        .SBTTL  INPCHR -- Move character to user's data buffer
                                ;-----------------------------------------------------------------------
                                ;  INPCHR is called to store a data character into the user's buffer
                                ;  associated with the current read request.
                                ;  If this causes the read request to be completed, the current read
                                ;  queue element is returned to the system.
                                ;
                                ;  Inputs:
                                ;   R2 = Character to be stored
                                ;   R5 = CL unit index number
                                ;
002700  010446                  INPCHR: MOV     R4,-(SP)
                                ;
                                ;  Get address of current read queue element
                                ;
002702  016504  0000000         1$:     MOV     CL$RQH(R5),R4   ;Get pointer to current read queue element
002706  001410                          BEQ     9$              ;Br if no read request is pending
                                ;
                                ;  Store character into data buffer
                                ;
002710  010246                          MOV     R2,-(SP)        ;Stack the data char for PTBYT
002712  004737  0000000                 CALL    PTBYT           ;Move char to user's data buffer
                                ;
                                ;  Decrement remaining byte count and see if this completes the read request
                                ;
002716  005364  000000C                 DEC     Q.WCNT-Q.BLKN(R4);Does this complete the read request?
002722  001002                          BNE     9$              ;Br if not
                                ;
                                ;  The read request is completed.
                                ;  Return the queue element to the system.
                                ;
002724  004737  002734'                 CALL    RDFIN           ;Read request is completed
                                ;
                                ;  Finished
                                ;
002730  012604                  9$:     MOV     (SP)+,R4
002732  000207                          RETURN
```

```
                                        .SBTTL  RDFIN  -- Completed a read request
                                ;-----------------------------------------------------------------------
                                ;  We have completed a read request.
                                ;  Null fill the remainder of the user's buffer if that is needed and then
                                ;  call the system I/O completion routine.
                                ;
                                ;  Inputs:
                                ;   R5 = CL unit index number.
                                ;
002734  010346                 RDFIN:  MOV     R3,-(SP)
002736  010446                         MOV     R4,-(SP)
                                ;
                                ;  Get address of current read queue element
                                ;
002740  016504  0000000                 MOV     CL$RQH(R5),R4   ;Get address of read queue element
002744  001427                          BEQ     9$              ;Br if none pending
                                ;
                                ;  See if we need to store nulls into the remainder of the buffer
                                ;
002746  016403  000000C                 MOV     Q.WCNT-Q.BLKN(R4),R3 ;Get remaining byte count
002752  001404                          BEQ     2$              ;Br if buffer is full
002754  005046                 1$:     CLR     -(SP)
002756  004737  0000000               CALL    PTBYT           ;Null fill the remainder of the buffer
002762  077304                          SOB     R3,1$
                                ;
                                ;  Remove the queue element from our internal queue and place on the queue
                                ;  of elements waiting to be returned to the system.
                                ;
002764                          2$:     DISABL                  ;;;** Disable interrupts **
002772  016465  000000C 0000000         MOV     Q.LINK-Q.BLKN(R4),CL$RQH(R5) ;;;Remove Q element from list
003000  013764  000004' 000000C         MOV     CQH,Q.LINK-Q.BLKN(R4) ;;;Put Q element on completion list
003006  010437  000004'                 MOV     R4,CQH
003012                                  ENABL                   ;;;  ** Enable interrupts **
                                ;
                                ;  Now call system I/O completion routine to free the queue element
                                ;
003020  004737  006002'                 CALL    RTNQ            ;Return queue element to the system
                                ;
                                ;  Finished
                                ;
003024  012604                  9$:     MOV     (SP)+,R4
003026  012603                          MOV     (SP)+,R3
003030  000207                          RETURN
```

```
                                        .SBTTL  CLTIMR -- Routine called from clock interrupt routine
                                ;-----------------------------------------------------------------------
                                ;  CLTIMR is called on a clock interrupt (50/60 Hz) basis to move characters
                                ;  to/from the user's I/O data buffer and the output/input CL character
                                ;  ring buffers.  We do this type of processing on a clock interrupt
                                ;  basis to avoid having to do a .FORK on each input/output character
                                ;  interrupt.
                                ;
003032  010146                  CLTIMR: MOV     R1,-(SP)
003034  010446                          MOV     R4,-(SP)
003036  010546                          MOV     R5,-(SP)
                                ;
                                ;  Begin loop to service each CL unit
                                ;
003040  012705  000000C                 MOV     #2*<CLTOTL-1>,R5;Get index # of last CL unit
                                ;
                                ;  See if this CL unit is connected to a line
                                ;
003044  016501  0000000         1$:     MOV     CL$LIX(R5),R1   ;Is this CL unit connected to a line?
003050  001412                          BEQ     2$              ;Br if not
                                ;
                                ;  See if user wants to change status of Data Terminal Ready
                                ;
003052  004737  006242'                 CALL    SETDTR          ;Call routine to set or clear the DTR flag
                                ;
                                ;  Call ORINGP for each line to try to move characters from the user's buffer
                                ;  to the output ring buffer.
                                ;
003056  005765  0000000                 TST     CL$XLN(R5)      ;Is this CL unit cross connected to TT line?
003062  001403                          BEQ     3$              ;Br if not
003064  004737  004566'                 CALL    CLOCPY          ;Copy characters to CL output ring buffer
003070  000402                          BR      2$
003072  004737  003122'         3$:     CALL    ORINGP          ;Move chars to output ring buffer
                                ;
                                ;  Process the next CL unit
                                ;
003076                          2$:     ENABL                   ;Make sure interrupts are enabled
003104  162705  000002                  SUB     #2,R5           ;Get index of next line
003110  002355                          BGE     1$              ;Loop if more lines to service
                                ;
                                ;  Finished
                                ;
003112  012605                          MOV     (SP)+,R5
003114  012604                          MOV     (SP)+,R4
003116  012601                          MOV     (SP)+,R1
003120  000207                          RETURN
```

```
                                        .SBTTL  ORINGP -- Move chars from data buffer to output ring buffer
                                ;-----------------------------------------------------------------------
                                ;  ORINGP is called to move characters from the current output data buffer
                                ;  to the output ring buffer.
                                ;
                                ;  Inputs:
                                ;   R5 = CL unit index number
                                ;
003122  010246                  ORINGP: MOV     R2,-(SP)
003124  010346                          MOV     R3,-(SP)
                                ;
                                ;  See if this routine is already being used by this unit.
                                ;  If so, don't reenter it (the other process will transfer all characters
                                ;  that can be transferred).
                                ;
003126                                  DISABL                          ;;;** Disable interrupts **
003134  032765  0000000 0000000         BIT     #CM$ORP,CL$STA(R5);;;Is this routine already active for unit?
003142  001402                          BEQ     21$                     ;;;Br if not
003144  000137  003562'                  JMP     9$                      ;;;Br if routine already active
                                ;
                                ;  This routine is not active for this unit.  Claim it.
                                ;
003150  052765  0000000 0000000 21$:    BIS     #CM$ORP,CL$STA(R5);;;Say routine is now active
003156                                  ENABL                           ;** Enable interrupts **
003164  005002                  11$:    CLR     R2                      ;Count # chars moved to output ring buffer
                                ;
                                ;  See if there is any free space in the output ring buffer and see if
                                ;  there is a pending write request for this unit.
                                ;
003166                          4$:     DISABL                          ;** Disable interrupts **
003174  005765  0000000                 TST     CL$ORS(R5)              ;;;Any available space in ring buffer?
003200  001555                          BEQ     8$                      ;;;Br if no space available
003202  005765  0000000                 TST     CL$WQH(R5)              ;;;Is there a pending write request?
003206  001004                          BNE     20$                     ;;;Br if a write is pending
003210  032765  0000000 0000000         BIT     #CM$EFP,CL$STA(R5);;;Are we doing end-of-file processing?
003216  001546                          BEQ     8$                      ;;;Br if not
                                ;
                                ;  There is free space in the output ring buffer and there is a pending
                                ;  write request.
                                ;  We will move characters from the user's buffer to the output ring buffer.
                                ;
003220                          20$:    ENABL                           ;** Enable interrupts **
                                ;
                                ;  See if we are sending spaces to simulate tabs
                                ;
003226  032765  0000000 0000000 15$:    BIT     #CM$TBS,CL$STA(R5);Are we doing tab simulation?
003234  001412                          BEQ     16$                     ;Br if not
003236  032765  000007  0000000         BIT     #7,CL$COL(R5)           ;Have we reached the next tab stop?
003244  001403                          BEQ     2$                      ;Br if yes
003246  012700  000040                  MOV     #SPACE,R0               ;Get space for simulation
003252  000474                          BR      12$
003254  042765  0000000 0000000 2$:     BIC     #CM$TBS,CL$STA(R5);Say we are finished with tab simulation
                                ;
                                ;  See if we are sending line feeds to simulate a form feed
                                ;
003262  032765  0000000 0000000 16$:    BIT     #CM$FFS,CL$STA(R5);Are we doing form feed simulation?
003270  001414                          BEQ     1$                      ;Br if not
```

```
     58 003272  026565  0000000 0000000          CMP     CL$LIN(R5),CL$LEN(R5) ;Have we reached top of new page yet?
     59 003300  103003                          BHIS    17$                   ;Br if yes
     60 003302  012700  000012                  MOV     #LF,R0                ;Send a line feed
     61 003306  000467                          BR      7$                    ;Go process the line feed
     62 003310  042765  0000000 0000000 17$:    BIC     #CM$FFS,CL$STA(R5);Say we have finished form feed simulation
     63 003316  005065  0000000                 CLR     CL$LIN(R5)            ;Say we are at top of new page
     64                                     ;
     65                                     ;  Try to get next character from user's data buffer
     66                                     ;
     67 003322  004737  003576' 1$:         CALL    GETCHR                ;Get next char from user's data buffer
     68 003326  103717                          BCS     4$                    ;Br if no chars left
     69                                     ;
     70                                     ;  Ignore user's FF immediately following FF from skip
     71                                     ;
     72 003330  032765  0000000 0000000          BIT     #CM$FFI,CL$STA(R5) ;Did we just do skip and should ignore FF?
     73 003336  001406                          BEQ     13$                   ;Br if not
     74 003340  042765  0000000 0000000          BIC     #CM$FFI,CL$STA(R5) ;Only ignore the 1st one
     75 003346  020027  000014                  CMP     R0,#FF                ;Is the 1st char after skip an FF?
     76 003352  001725                          BEQ     15$                   ;If yes, ignore this char
     77                                     ;
     78                                     ;  See if this is a control character
     79                                     ;
     80 003354  032765  0000000 0000000 13$:    BIT     #CO$BNO,CL$OPT(R5);Are we in binary output mode?
     81 003362  001046                          BNE     5$                    ;Br if yes -- Accept all chars
     82 003364  032765  0000000 0000000          BIT     #CO$8BT,CL$OPT(R5);Is 8 bit support wanted?
     83 003372  001002                          BNE     18$                   ;Br if yes
     84 003374  042700  177600                  BIC     #^C<177>,R0           ;Mask character to 7 bits
     85 003400  020027  000037          18$:    CMP     R0,#37                ;Is this a control character?
     86 003404  101430                          BLOS    7$                    ;Br if yes
     87 003406  042765  0000000 0000000          BIC     #CM$CRL,CL$STA(R5) ;Remember this is not a carriage return
     88                                     ;
     89                                     ;  This is not a control character.
     90                                     ;  See if we should translate lower-case to upper-case
     91                                     ;
     92 003414  032765  0000000 0000000          BIT     #CO$LC,CL$OPT(R5);May we send lower-case characters?
     93 003422  001010                          BNE     12$                   ;Br if yes
     94 003424  020027  000141                  CMP     R0,#141               ;Is this a lower-case letter?
     95 003430  103405                          BLO     12$                   ;Br if not
     96 003432  120027  000172                  CMPB    R0,#172
     97 003436  101002                          BHI     12$
     98 003440  162700  000040                  SUB     #40,R0                ;Convert lower-case to upper case
     99                                     ;
    100                                     ;  See if we need to truncate line due to WIDTH parameter
    101                                     ;
    102 003444  005265  0000000         12$:    INC     CL$COL(R5)            ;Advance column counter
    103 003450  016503  0000000                 MOV     CL$WID(R5),R3         ;Was a WIDTH parameter specified?
    104 003454  001411                          BEQ     5$                    ;Br if not
    105 003456  026503  0000000                 CMP     CL$COL(R5),R3         ;Have we reached the specified width?
    106 003462  101406                          BLOS    5$                    ;Br if not
    107 003464  000660                          BR      15$                   ;Discard this char if line is too wide
    108                                     ;
    109                                     ;  This is a control character.
    110                                     ;  Call control character processing routine.
    111                                     ;
    112 003466  010003                  7$:     MOV     R0,R3                 ;Get control character
    113 003470  006303                          ASL     R3                    ;Convert to word table index
    114 003472  004773  004044'                 CALL    @CCORTN(R3)           ;Call processing routine
```

```
003476  103653                          BCS     15$             ;Br if we should discard this character
                                ;
                                ;  Move character to output ring buffer
                                ;
003500  016503  0000000         5$:     MOV     CL$ORP(R5),R3   ;Get position for char in ring buffer
003504  110023                          MOVB    R0,(R3)+        ;Store char into ring buffer
                                ;
                                ;  Say 1 less free char space in ring buffer
                                ;
003506  005365  0000000                 DEC     CL$ORS(R5)      ;One less free char pos in out ring buffer
003512  005202                          INC     R2              ;Count # chars moved to ring buffer
                                ;
                                ;  Save updated ring buffer pointer
                                ;
003514  020365  0000000                 CMP     R3,CL$ORE(R5)   ;Did we advance past end of ring buffer?
003520  103402                          BLO     6$              ;Br if not
003522  016503  0000000                 MOV     CL$ORB(R5),R3   ;Wrap around to front of ring buffer
003526  010365  0000000         6$:     MOV     R3,CL$ORP(R5)   ;Save new ring buffer pointer
003532  000615                          BR      4$              ;Go see if we should send more chars
                                ;
                                ;  Finished moving characters to output ring buffer.
                                ;  If we moved any characters, call the routine to try to start output
                                ;  to the line.
                                ;
003534  005702                  8$:     TST     R2              ;;;Did we move any characters to ring buffer?
003536  001406                          BEQ     10$             ;;;Br if not
003540                                  ENABL                   ;** Enable interrupts **
003546  004737  005506'                 CALL    CLSTRT          ;Try to start transmission to this line
003552  000604                          BR      11$             ;Go back and check for more to send
                                ;
                                ;  Release this routine for this unit
                                ;
003554  042765  0000000 0000000 10$:     BIC     #CM$ORP,CL$STA(R5);;;Say routine is now free
                                ;
                                ;  Finished
                                ;
003562                          9$:     ENABL                   ;** Enable interrupts **
003570  012603                          MOV     (SP)+,R3
003572  012602                          MOV     (SP)+,R2
003574  000207                          RETURN
```

```
                                        .SBTTL  GETCHR -- Get next output char from user's data buffer
                                ;-----------------------------------------------------------------------
                                ;  GETCHR is called to obtain the next character from the user's
                                ;  data buffer.
                                ;
                                ;  Inputs:
                                ;   R5 = CL unit index number
                                ;
                                ;  Outputs:
                                ;   C-flag cleared ==> A character was gotten
                                ;   C-flag set     ==> No more characters are available
                                ;   R0 = Character gotten if C-flag is cleared
                                ;
003576  010446                  GETCHR: MOV     R4,-(SP)
                                ;
                                ;  See if we should do end-of-file output processing.
                                ;
003600  032765  0000000 0000000 5$:     BIT     #CM$EFP,CL$STA(R5) ;Should we do end-of-file processing?
003606  001403                          BEQ     2$                 ;Br if not
                                ;
                                ;  We are doing end-of-file output processing.
                                ;  See if there is another end-of-file character to send.
                                ;
003610  004737  003752'                 CALL    EOFCHR             ;See if another eof char to send
003614  103054                          BCC     12$                ;Br if we got an EOF character
                                ;
                                ;  See if there is a pending write operation
                                ;
003616  016504  0000000         2$:     MOV     CL$WQH(R5),R4      ;Get pointer to current write queue element
003622  001446                          BEQ     10$                ;Br if no pending write operation
                                ;
                                ;  If the FORMO option is in effect and this is the first write to
                                ;  block 0, send a form feed.
                                ;
003624  005764  000000C                 TST     Q.BLKN-Q.BLKN(R4);Is block number = 0?
003630  001011                          BNE     4$                 ;Br if not
003632  032765  0000000 0000000         BIT     #CO$FFO,CL$OPT(R5) ;Is the FORMO option in effect?
003640  001405                          BEQ     4$                 ;Br if not
003642  005264  000000C                 INC     Q.BLKN-Q.BLKN(R4);Inc block # so we only do this once
003646  112700  000014                  MOVB    #FF,R0             ;Get form feed character
003652  000434                          BR      9$                 ;Return the form feed
                                ;
                                ;  See if current queue element has another character to be sent
                                ;
003654  005764  000000C         4$:     TST     Q.WCNT-Q.BLKN(R4);Any remaining bytes to send?
003660  001406                          BEQ     3$                 ;Br if not -- write request is finished
                                ;
                                ;  Get next character from user's buffer
                                ;
003662  005364  000000C                 DEC     Q.WCNT-Q.BLKN(R4);Decrease remaining byte count
003666  004737  0000000                 CALL    GTBYT              ;Get next byte from user's buffer
003672  012600                          MOV     (SP)+,R0           ;Get the returned character
003674  000423                          BR      9$                 ;Return the character
                                ;
                                ;  This write operation is completed.
                                ;  Remove the queue element from our internal queue and place it
                                ;  on the queue of elements waiting to be returned to the system.
```

```
                                        ;
003676                                  3$:     DISABL                  ;;;** Disable interrupts **
003704  016465  000000C 000000G                MOV     Q.LINK-Q.BLKN(R4),CL$WQH(R5) ;;;Remove element from internal Q
003712  013764  000004' 000000C                MOV     CQH,Q.LINK-Q.BLKN(R4) ;;;Add to list of completed requests
003720  010437  000004'                        MOV     R4,CQH
003724                                          ENABL                   ;** Enable interrupts **
                                        ;
                                        ;  Return the completed queue element to the system (do .DRFIN)
                                        ;
003732  004737  006002'                         CALL    RTNQ            ;Tell system we finished the operation
                                        ;
                                        ;  Go back and see if there is another write request pending
                                        ;
003736  000727                                 BR      2$              ;Go check for another write request
                                        ;
                                        ;  There are no available characters
                                        ;
003740  000261                          10$:    SEC                     ;Signal that no chars are available
003742  000401                                  BR      12$
                                        ;
                                        ;  We got a character
                                        ;
003744  000241                          9$:     CLC                     ;Signal that we got a character
                                        ;
                                        ;  Finished
                                        ;
003746  012604                          12$:    MOV     (SP)+,R4
003750  000207                                  RETURN
```

```
                                        .SBTTL  EOFCHR -- Get next end-of-file output character
                                ;-----------------------------------------------------------------------
                                ;  This routine is called during end-of-file output processing to see
                                ;  if there is another end-of-file output character to send.
                                ;
                                ;  Inputs:
                                ;   R5 = CL unit index number
                                ;
                                ;  Outputs:
                                ;   C-flag cleared ==> Got a character
                                ;   C-flag set ==> No more characters
                                ;   R0 = Character gotten if C-flag cleared.
                                ;
003752                          EOFCHR:
                                ;
                                ;  See if we need to send form-feeds
                                ;
003752  126565  0000010 0000000         CMPB    CL$EPN+1(R5),CL$EPN(R5);Do we need to send more form-feeds?
003760  103005                         BHIS    1$                      ;Br if not
003762  105265  0000010                 INCB    CL$EPN+1(R5)            ;Count another form-feed being sent
003766  012700  000014                  MOV     #FF,R0                  ;Get form-feed character
003772  000422                          BR      7$                      ;Go send it
                                ;
                                ;  See if we need to send characters from ENDSTRING
                                ;
003774  016500  0000000         1$:     MOV     CL$EPP(R5),R0           ;Are we sending end-string characters?
004000  001405                         BEQ     2$                      ;Br if not
004002  111000                         MOVB    (R0),R0                 ;Get next char to send
004004  001403                         BEQ     2$                      ;Br if reached end of string
004006  005265  0000000                 INC     CL$EPP(R5)              ;Advance character pointer
004012  000412                          BR      7$                      ;Go send the character
                                ;
                                ;  We have finished all end-of-file output processing
                                ;
004014  105065  0000010         2$:     CLRB    CL$EPN+1(R5)            ;Reset form-feed count
004020  016565  0000000 0000000         MOV     CL$EPS(R5),CL$EPP(R5);Reset end-string pointer
004026  042765  0000000 0000000         BIC     #CM$EFP,CL$STA(R5);Finished end-of-file output processing
004034  000261                          SEC                             ;Signal that no character was gotten
004036  000401                          BR      9$
                                ;
                                ;  We got a character
                                ;
004040  000241                  7$:     CLC                             ;Signal that we got a character
                                ;
                                ;  Finished
                                ;
004042  000207                  9$:     RETURN
```

```
                                        .SBTTL  CCORTN -- Output control character processing routines
                                ;-----------------------------------------------------------------------
                                ;  Processing routines for output control characters.
                                ;  When one of these routines is called, R0 contains the control character.
                                ;  If the character is to be sent, the C-flag is cleared on return.
                                ;  If the character is to be discarded, the C-flag is set on return.
                                ;
                                ;  Vector of control character processing routines
                                ;
004044  004172'         CCORTN: .WORD   CCONUL           ;00 null
004046  004144'                 .WORD   CCOCTL           ;01 SHO
004050  004144'                 .WORD   CCOCTL           ;02 STX
004052  004144'                 .WORD   CCOCTL           ;03 ETX
004054  004144'                 .WORD   CCOCTL           ;04 EOT
004056  004144'                 .WORD   CCOCTL           ;05 ENQ
004060  004144'                 .WORD   CCOCTL           ;06 ACK
004062  004144'                 .WORD   CCOCTL           ;07 BEL
004064  004176'                 .WORD   CCOBS            ;10 BACKSPACE
004066  004220'                 .WORD   CCOTAB           ;11 TAB
004070  004312'                 .WORD   CCOLF            ;12 LINE FEED
004072  004144'                 .WORD   CCOCTL           ;13 VT
004074  004446'                 .WORD   CCOFF            ;14 FF
004076  004522'                 .WORD   CCOCR            ;15 CARRIAGE RETURN
004100  004144'                 .WORD   CCOCTL           ;16 SO
004102  004144'                 .WORD   CCOCTL           ;17 SI
004104  004144'                 .WORD   CCOCTL           ;20 DLE
004106  004144'                 .WORD   CCOCTL           ;21 DC1 (ctrl-Q)
004110  004144'                 .WORD   CCOCTL           ;22 DC2
004112  004144'                 .WORD   CCOCTL           ;23 DC3 (ctrl-S)
004114  004144'                 .WORD   CCOCTL           ;24 DC4
004116  004144'                 .WORD   CCOCTL           ;25 NAK
004120  004144'                 .WORD   CCOCTL           ;26 SYN
004122  004144'                 .WORD   CCOCTL           ;27 ETB
004124  004144'                 .WORD   CCOCTL           ;30 CAN
004126  004144'                 .WORD   CCOCTL           ;31 EM
004130  004144'                 .WORD   CCOCTL           ;32 SUB (ctrl-Z)
004132  004144'                 .WORD   CCOCTL           ;33 ESC
004134  004144'                 .WORD   CCOCTL           ;34 FS
004136  004144'                 .WORD   CCOCTL           ;35 GS
004140  004144'                 .WORD   CCOCTL           ;36 RS
004142  004144'                 .WORD   CCOCTL           ;37 US
```

```
                                        ;
                                        ;  Process a general control character
                                        ;
004144  042765  000000@ 000000@ CCOCTL: BIC     #CM$CRL,CL$STA(R5) ;Say last char out was not carriage return
004152  032765  000000@ 000000@        BIT     #CO$CTL,CL$OPT(R5) ;Are we to transmit control chars?
004160  001002                       BNE     CCOSND             ;Br if yes
004162  000261                       SEC                        ;Say to ignore this character
004164  000207                       RETURN
                                        ;
                                        ;  Routine to cause the current control character to be transmitted unchanged
                                        ;
004166  000241                       CCOSND: CLC                ;Say to send the character
004170  000207                       RETURN
                                        ;
                                        ;  Process null character
                                        ;
004172  000261                       CCONUL: SEC                ;Say to ignore this character
004174  000207                       RETURN
                                        ;
                                        ;  Proccess Backspace character
                                        ;
004176  042765  000000@ 000000@ CCOBS:  BIC     #CM$CRL,CL$STA(R5) ;Say last char out was not carriage return
004204  005365  000000@                DEC     CL$COL(R5)         ;Say we are moving back 1 char
004210  002366                       BGE     CCOSND             ;Br if did not go past column 0
004212  005065  000000@                CLR     CL$COL(R5)         ;Constrain to column 0
004216  000763                       BR      CCOSND             ;Go send the character
                                        ;
                                        ;  Process tab character
                                        ;
004220  042765  000000@ 000000@ CCOTAB: BIC     #CM$CRL,CL$STA(R5) ;Say last char out was not carriage return
004226  032765  000000@ 000000@        BIT     #CO$TAB,CL$OPT(R5) ;Does device have hardware tab support
004234  001416                       BEQ     1$                 ;Br if not
004236  062765  000010  000000@        ADD     #8.,CL$COL(R5)     ;Bound up to next tab stop
004244  042765  000007  000000@        BIC     #7,CL$COL(R5)
004252  005765  000000@                TST     CL$WID(R5)         ;Was a maximum width specified?
004256  001743                       BEQ     CCOSND             ;Br if not -- go send the tab
004260  026565  000000@ 000000@        CMP     CL$COL(R5),CL$WID(R5) ;Have we gone beyond max width?
004266  103737                       BLO     CCOSND             ;Br if not
004270  000740                       BR      CCONUL             ;Discard this tab
004272  052765  000000@ 000000@ 1$:    BIS     #CM$TBS,CL$STA(R5) ;Say we are doing tab simulation
004300  005265  000000@                INC     CL$COL(R5)         ;Advance column counter
004304  012700  000040                MOV     #SPACE,R0          ;Send a space character
004310  000726                       BR      CCOSND
                                        ;
                                        ;  Process Line feed character
                                        ;
004312  005265  000000@          CCOLF: INC     CL$LIN(R5)         ;Increment line-on-page counter
004316  016500  000000@                MOV     CL$LEN(R5),R0      ;Was a page length value specified?
004322  001431                       BEQ     5$                 ;Br if not
004324  026500  000000@                CMP     CL$LIN(R5),R0      ;Have we reached the top of a new page?
004330  103405                       BLO     2$                 ;Br if not
004332  005065  000000@                CLR     CL$LIN(R5)         ;Say we are at top of a new page
004336  042765  000000@ 000000@        BIC     #CM$FFS,CL$STA(R5);Stop doing form feed simulation
004344  166500  000000@          2$:   SUB     CL$SKP(R5),R0      ;See if we are to skip lines at bottom of page
004350  026500  000000@                CMP     CL$LIN(R5),R0      ;Have we reached the skip point?
004354  001014                       BNE     5$                 ;Br if not
004356  032765  000000@ 000000@        BIT     #CM$FFS,CL$STA(R5);Are we already doing form feed simulation?
```

```
                        001010                          BNE     5$                      ;Br if yes
                        112700  000014                  MOVB    #FF,R0                  ;At skip point -- Do a form feed
                        052765  000000@ 000000@         BIS     #CM$FFI,CL$STA(R5) ;Ignore FF if 1st char after skip
                        005365  000000@                 DEC     CL$LIN(R5)              ;Set line counter back -- haven't sent LF yet
                        000420                          BR      CCOFF                   ;Go process the form feed
                        032765  000000@ 000000@ 5$:     BIT     #CO$LFO,CL$OPT(R5) ;Should we discard line feeds on output?
                        001010                          BNE     6$                      ;Br if not
                        032765  000000@ 000000@         BIT     #CM$CRL,CL$STA(R5);Was last char out a carriage return?
                        001404                          BEQ     6$                      ;Br if not
                        042765  000000@ 000000@         BIC     #CM$CRL,CL$STA(R5);Clear flag that says carriage return last
                        000656                          BR      CCONUL                  ;Discard the line feed
                        112700  000012          6$:     MOVB    #LF,R0                  ;Get back line feed character
                        000241                          CLC                             ;Say to send it
                        000207                  9$:     RETURN
                                                ;
                                                ;  Process Form feed character
                                                ;
                        042765  000000@ 000000@ CCOFF:  BIC     #CM$CRL,CL$STA(R5) ;Say last char out was not carriage return
                        032765  000000@ 000000@         BIT     #CO$FF,CL$OPT(R5) ;Does this device support form feed chars?
                        001403                          BEQ     1$                      ;Br if not
                        005065  000000@                 CLR     CL$LIN(R5)              ;Say we are at top of the page
                        000636                          BR      CCOSND                  ;Go send the form feed
                        005765  000000@         1$:     TST     CL$LEN(R5)              ;Do we have a non-zero page length?
                        001406                          BEQ     2$                      ;If not then discard the FF
                        052765  000000@ 000000@         BIS     #CM$FFS,CL$STA(R5);Say we are starting form-feed simulation
                        012700  000012                  MOV     #LF,R0                  ;Translate form feed to line feed
                        000677                          BR      CCOLF                   ;Go send line feed
                        005065  000000@         2$:     CLR     CL$LIN(R5)              ;Say we are at top of page
                        000624                          BR      CCONUL                  ;Discard the character
                                                ;
                                                ;  Process carriage return character
                                                ;
                        052765  000000@ 000000@ CCOCR:  BIS     #CM$CRL,CL$STA(R5) ;Say last char out was carriage return
                        005065  000000@                 CLR     CL$COL(R5)              ;Say we are back to column 0
                        032765  000000@ 000000@         BIT     #CO$CR,CL$OPT(R5);Should we transmit carriage returns?
                        001211                          BNE     CCOSND                  ;Br if yes
                        000261                          SEC                             ;Ignore this char
                        000207                          RETURN
```

```
                                        .SBTTL  CLXICP -- Got char for output to cross connected CL line
                                ;-----------------------------------------------------------------------
                                ;  CLXICP is called at fork level when a character is received from a
                                ;  TT line that is cross connected to a CL line.
                                ;  It copies all possible characters from the input silo of the TT line
                                ;  to the output silo for the CL line and initiates output to the CL line.
                                ;
                                ;  Inputs:
                                ;   R1 = Index number of TT line that received the character
                                ;
004550  010546                  CLXICP: MOV     R5,-(SP)
                                ;
                                ;  Get CL index of line we are cross connected to
                                ;
004552  016105  0000006                 MOV     LXCL(R1),R5     ;Get # of CL line we are connected to
                                ;
                                ;  Call routine to copy all chars from TT input silo to CL output silo
                                ;
004556  004737  004566'                 CALL    CLOCPY          ;Copy chars to CL output silo
                                ;
                                ;  Finished
                                ;
004562  012605                          MOV     (SP)+,R5
004564  000207                          RETURN
```

```
                                        .SBTTL  CLOCPY -- Copy characters from TT input buf to CL output buf
                                ;-----------------------------------------------------------------------
                                ;  CLOCPY is called to copy characters from the input silo of a TT line to
                                ;  the output buffer of a cross-connected CL line.
                                ;
                                ;  Inputs:
                                ;   R5 = Unit index of CL line
                                ;
004566  010146                  CLOCPY: MOV     R1,-(SP)
004570  010246                          MOV     R2,-(SP)
004572  010346                          MOV     R3,-(SP)
                                ;
                                ;  See if this routine is already being used by this unit.
                                ;  If so, don't reenter it (the other process will transfer all
                                ;  characters that can be transferred).
                                ;
004574                                  DISABL                          ;;;Disable interrupts
004602  032765  000000G 000000G          BIT     #CM$ORP,CL$STA(R5);;;Is this routine already active?
004610  001113                          BNE     9$                      ;;;Br if yes
                                ;
                                ;  This routine is not active for this unit.  Claim it.
                                ;
004612  052765  000000G 000000G          BIS     #CM$ORP,CL$STA(R5);;;Say routine is now active
004620                                  ENABL                           ;Enable interrupts
004626  005002                  11$:    CLR     R2                      ;Count # chars copied to output buffer
                                ;
                                ;  See if cross-connection is still in effect
                                ;
004630  016501  000000G         4$:     MOV     CL$XLN(R5),R1           ;Get number of cross-connected TT line
004634  001476                          BEQ     10$                     ;Br if no longer cross connected
                                ;
                                ;  See if there is any free space in the output ring buffer.
                                ;
004636                                  DISABL                          ;;;Disable interrupts
004644  005765  000000G                 TST     CL$ORS(R5)              ;;;Any available space in ring buffer?
004650  001460                          BEQ     8$                      ;;;Br if no space available
004652  026161  000000G 000000G          CMP     LHIRBS(R1),LHIRBA(R1);;;Any chars in TT input silo?
004660  001454                          BEQ     8$                      ;;;Br if not
004662                                  ENABL                           ;Enable interrupts
                                ;
                                ;  Get next character from TT input silo
                                ;
004670  004777  000000G                 CALL    @SILFET                 ;Get next char from TT input silo
004674  103446                          BCS     8$                      ;Br if no more chars available
                                ;
                                ;  We got a character.
                                ;  See if character has special significance.
                                ;
004676  032765  000000G 000000G          BIT     #CM$MCC,CL$STA(R5);Modem control or literal char?
004704  001407                          BEQ     1$                      ;Br if not
004706  042765  000000G 000000G          BIC     #CM$MCC,CL$STA(R5);Reset literal-character flag
004714  004737  005056'                 CALL    CLXMCC                  ;Process the character
004720  103743                          BCS     4$                      ;Br if finished with char
004722  000415                          BR      2$                      ;Go transmit the character
004724  120037  000000G         1$:     CMPB    R0,VCXTRM               ;Control-\ -- Terminate connection?
004730  001003                          BNE     3$                      ;Br if not
004732  004737  005404'                 CALL    CLXBRK                  ;Break cross connection and drop DTR
```

```
 58 004736  000425                          BR      8$                  ;Finished
 59 004740  120037  0000000          3$:     CMPB    R0,VCXCTL           ;Control-A means next char is modem control
 60 004744  001004                          BNE     2$                  ;Br if not ctrl-A
 61 004746  052765  0000000 0000000         BIS     #CM$MCC,CL$STA(R5);Remember next char is modem control
 62 004754  000725                          BR      4$                  ;Go get next char
 63                                 ;
 64                                 ;  Store this character into the output ring buffer
 65                                 ;
 66 004756  016503  0000000          2$:     MOV     CL$ORP(R5),R3       ;Get position for char in ring buffer
 67 004762  110023                          MOVB    R0,(R3)+            ;Store char into ring buffer
 68                                 ;
 69                                 ;  Count chars in ring buffer
 70                                 ;
 71 004764  005202                          INC     R2                  ;One more char stored into ring buffer
 72 004766  005365  0000000                 DEC     CL$ORS(R5)          ;One less free space in ring buffer
 73                                 ;
 74                                 ;  Save updated ring buffer pointer
 75                                 ;
 76 004772  020365  0000000                 CMP     R3,CL$ORE(R5)       ;Did we advance past end of ring buffer?
 77 004776  103402                          BLO     6$                  ;Br if not
 78 005000  016503  0000000                 MOV     CL$ORB(R5),R3       ;Wrap around to front of ring buffer
 79 005004  010365  0000000          6$:     MOV     R3,CL$ORP(R5)       ;Save new ring buffer pointer
 80 005010  000707                          BR      4$                  ;Go see if we have more chars to move
 81                                 ;
 82                                 ;  We have copied all the characters we can from the TT input silo
 83                                 ;  buffer to the CL output ring buffer.
 84                                 ;  If we copied any characters, call the routine to try to start
 85                                 ;  output for the CL line.
 86                                 ;
 87 005012  005702                  8$:     TST     R2                  ;;;Did we copy any characters?
 88 005014  001406                          BEQ     10$                 ;;;Br if not
 89 005016                                  ENABL                       ;Enable interrupts
 90 005024  004737  005506'                 CALL    CLSTRT              ;Start transmission to CL line
 91 005030  000676                          BR      11$                 ;Go back and try to copy more
 92                                 ;
 93                                 ;  Release this routine for this unit
 94                                 ;
 95 005032  042765  0000000 0000000 10$:     BIC     #CM$ORP,CL$STA(R5);Say routine is now free
 96                                 ;
 97                                 ;  Finished
 98                                 ;
 99 005040                          9$:     ENABL                       ;Enable interrupts
100 005046  012603                          MOV     (SP)+,R3
101 005050  012602                          MOV     (SP)+,R2
102 005052  012601                          MOV     (SP)+,R1
103 005054  000207                          RETURN
```

```
                                        .SBTTL  CLXMCC -- Process cross connect modem control character
                                ;-----------------------------------------------------------------------
                                ;  Process a modem control character for a cross connection.
                                ;
                                ;  Inputs:
                                ;   R0 = Character
                                ;   R5 = CL unit index number
                                ;
                                ;  Outputs:
                                ;   C-flag cleared ==> Go ahead and transmit this character.
                                ;   C-flag set     ==> Do not transmit this character.
                                ;
005056  010046                  CLXMCC: MOV     R0,-(SP)        ;Save the character
005060  010146                          MOV     R1,-(SP)
                                ;
                                ;  Translate lower-case to upper-case
                                ;
005062  120027  000141                  CMPB    R0,#141         ;Is this a lower-case letter?
005066  103405                          BLO     1$              ;Br if not
005070  120027  000172                  CMPB    R0,#172
005074  101002                          BHI     1$              ;Br if not
005076  162700  000040                  SUB     #40,R0          ;Convert to upper-case
                                ;
                                ;  "B" -- Start sending a break
                                ;
005102  120027  000102          1$:     CMPB    R0,#'B          ;Is character B?
005106  001042                          BNE     2$              ;Br if not
005110  052765  000000G 000000G        BIS     #CM$BRK,CL$STA(R5);Set flag saying we are sending break
005116  012700  000000G                 MOV     #MS$BRK,R0      ;Set flag to start break transmission
005122  004737  006340'                 CALL    SETBRK          ;Call hardware routine to start sending break
005126  004737  005506'                 CALL    CLSTRT          ;Start transmitter
005132  004737  000000G                 CALL    GETRTQ          ;Get a real-time queue element (ptr in R1)
005136  012761  000036  000000G        MOV     #30.,CQ$LOT(R1) ;Set approx 0.5 second time interval
005144  012761  005354' 000000G        MOV     #CLXSSB,CQ$RTN(R1);Set address of compl routine
005152  013761  000000G 000000G        MOV     @#KPAR5,CQ$PA5(R1);Save system par 5 mapping
005160  010561  000000G                 MOV     R5,CQ$RO(R1)    ;Set CL unit index
005164                                  DISABL                  ;;;* Disable interrupts *
005172  013761  000000G 000000G        MOV     MRKTHD,CQ$LNK(R1);;;Put new element on linked list
005200  010137  000000G                 MOV     R1,MRKTHD       ;;;
005204                                  ENABL                   ;* Enable interrupts *
005212  000452                          BR      20$
                                ;
                                ;  "D" -- Raise DTR
                                ;
005214  120027  000104          2$:     CMPB    R0,#'D          ;Is character D?
005220  001006                          BNE     4$              ;Br if not
005222  052765  000000G 000000G        BIS     #CO$DTR,CL$OPT(R5);Request DTR up
005230  004737  006242'                 CALL    SETDTR          ;Call routine to raise DTR
005234  000441                          BR      20$
                                ;
                                ;  "H" -- Drop DTR
                                ;
005236  120027  000110          4$:     CMPB    R0,#'H          ;Is character H?
005242  001006                          BNE     5$              ;Br if not
005244  042765  000000G 000000G        BIC     #CO$DTR,CL$OPT(R5);Request DTR drop
005252  004737  006242'                 CALL    SETDTR          ;Call routine to drop DTR
005256  000430                          BR      20$
```

```
                                        ;
                                        ;  "R" -- Reset XON/XOFF status
                                        ;
005260  120027  000122          5$:     CMPB    R0,#'R          ;Reset XON/XOFF status?
005264  001017                          BNE     6$              ;Br if not
005266  016501  0000000                 MOV     CL$LIX(R5),R1   ;Get index of line we are connected to
005272  042761  0000000 0000000         BIC     #$CTRLS,LSW3(R1);Reset XOFF received flag
005300  042761  0000000 0000000         BIC     #$HISTP,LSW10(R1);Say input has not been stopped by XOFF
005306  016100  0000000                 MOV     LCDTYP(R1),R0   ;Get device type code
005312  004770  0000000                 CALL    @CDSXON(R0)     ;Call routine to stuff XON into output
005316  004737  005506'                 CALL    CLSTRT          ;Try to start output to CL unit
005322  000406                          BR      20$
                                        ;
                                        ;  "X" -- Break cross connection without dropping DTR
                                        ;
005324  120027  000130          6$:     CMPB    R0,#'X          ;Break cross-connection?
005330  001005                          BNE     21$             ;Br if not
005332  004737  005424'                 CALL    CLXDRP          ;Break cross connection
005336  000400                          BR      20$             ;Finished with character
                                        ;
                                        ;  This is a modem control character.
                                        ;  Don't send it.
                                        ;
005340  000261                  20$:    SEC                     ;Signal not to send the character
005342  000401                          BR      22$
                                        ;
                                        ;  This is not a modem control character.
                                        ;  Send the character literally.
                                        ;
005344  000241                  21$:    CLC                     ;Signal that we should send the character
                                        ;
                                        ;  Finished
                                        ;
005346  012601                  22$:    MOV     (SP)+,R1
005350  012600                          MOV     (SP)+,R0
005352  000207                          RETURN
```

```
                                ;-----------------------------------------------------------------------
                                ;  System completion routine called to stop sending break to a
                                ;  cross-connected CL line.
                                ;
                                ;  Inputs:
                                ;   R0 = CL unit index
                                ;
005354  010546                  CLXSSB: MOV     R5,-(SP)
005356  010005                          MOV     R0,R5           ;Get CL unit index
                                ;
                                ;  Stop sending break
                                ;
005360  005000                          CLR     R0              ;Clear break-send flag
005362  004737  006340'                CALL    SETBRK          ;Call hardware routine to end break
005366  042765  0000000@ 0000000@       BIC     #CM$BRK,CL$STA(R5);Clear break-sending flag
005374  004737  005506'                 CALL    CLSTRT          ;Start transmitter
                                ;
                                ;  Finished
                                ;
005400  012605                          MOV     (SP)+,R5
005402  000207                          RETURN
```

```
                                        .SBTTL  CLXBRK -- Break a CL-TT cross connection and drop DTR
                                ;-----------------------------------------------------------------------
                                ;  This routine is called when we receive control-\ to break the cross
                                ;  connection between a CL unit and a TT line.
                                ;  In addition to breaking the connection, DTR is dropped to hang up.
                                ;
                                ;  Inputs:
                                ;   R5 = CL unit index
                                ;
005404                          CLXBRK:
                                ;
                                ;  First, drop DTR
                                ;
005404  042765  0000000 0000000         BIC     #CO$DTR,CL$OPT(R5) ;Request DTR drop
005412  004737  006242'                 CALL    SETDTR          ;Call routine to drop DTR
                                ;
                                ;  Now break the cross connection
                                ;
005416  004737  005424'                 CALL    CLXDRP          ;Break the cross connection
                                ;
                                ;  Finished
                                ;
005422  000207                          RETURN
```

```
                        .SBTTL  CLXDRP -- Break a CL-TT cross connection
                ;-----------------------------------------------------------------------
                ;  This routine is called when we receive control-\ to break the cross
                ;  connection between a CL unit and a TT line.
                ;  DTR is not dropped by this routine.  Call CLXBRK to drop DTR too.
                ;
                ;  Inputs:
                ;   R5 = CL unit index
                ;
005424  010146  CLXDRP: MOV     R1,-(SP)
                ;
                ;  Reset this CL unit
                ;
005426  004737  001756'         CALL    CLREST          ;Reset the CL unit
                ;
                ;  Reconnect time-sharing line to normal input character processing routine
                ;
005432                          DISABL                  ;;;** Disable interrupts **
005440  016501  0000000         MOV     CL$XLN(R5),R1   ;;;Get number of cross-connected TT line
005444  012761  177777  0000000 MOV     #-1,LXCL(R1)    ;;;Say not connected to a CL unit
005452  012761  0000000 0000000 MOV     #TTINCP,LINIR(R1);;;Connect to TT input processing routine
                ;
                ;  Say CL unit no longer connected to time-sharing line
                ;
005460  005065  0000000         CLR     CL$XLN(R5)      ;;;CL unit no longer connected to TT line
005464                          ENABL                   ;** Enable interrupts **
                ;
                ;  Restart the execution of the job
                ;
005472  116101  0000000         MOVB    LNMAP(R1),R1    ;Get virtual job index number
005476  004737  0000000         CALL    FORCEX          ;Cause job to continue execution
                ;
                ;  Finished
                ;
005502  012601                  MOV     (SP)+,R1
005504  000207                  RETURN
```

```
                                        .SBTTL  CLSTRT -- Start transmissions to a line
                                ;-----------------------------------------------------------------------
                                ;  CLSTRT is called to initiate transmission to a line.
                                ;
                                ;  Inputs:
                                ;   R5 = CL unit index number.
                                ;
005506  010146                  CLSTRT: MOV     R1,-(SP)
                                ;
                                ;  Convert CL unit number into line index number
                                ;
005510  016501  0000000                 MOV     CL$LIX(R5),R1   ;Get line index # for this CL unit
                                ;
                                ;  Call device dependent routine to start the transmitter
                                ;
005514  016100  0000000                 MOV     LCDTYP(R1),R0   ;Get communications device type code
005520  004770  0000000                 CALL    @CDSTRT(R0)     ;Call device dependent startup routine
005524  005237  0000000                 INC     NEDCDO          ;Say output character processing needed
005530  005237  0000000                 INC     NEDCLO          ;Say CL output processing needed
                                ;
                                ;  Finished
                                ;
005534  012601                          MOV     (SP)+,R1
005536  000207                          RETURN
```

```
                                        .SBTTL  CLABRT -- Handler abort routine
                                ;-----------------------------------------------------------------------
                                ;  CLABRT is jumped to from the handler abort entry point.
                                ;  It terminates any I/O operations for the job being aborted.
                                ;
                                ;  Inputs:
                                ;   R4 = Aborted job index number / 2
                                ;
005540  010346                  CLABRT: MOV     R3,-(SP)
005542  010446                          MOV     R4,-(SP)
005544  010546                          MOV     R5,-(SP)
                                ;
                                ;  Check each CL unit to see if there are any requests for this job
                                ;
005546  012705  000000C                 MOV     #2*<CLTOTL-1>,R5;Get index to last CL unit
005552  012703  0000000         1$:     MOV     #CL$RQH,R3      ;Get address of read queue head
005556  004737  005614'                 CALL    CKABTQ          ;See if there are any entries on this queue
005562  012703  0000000                 MOV     #CL$WQH,R3      ;Get address of write queue head
005566  004737  005614'                 CALL    CKABTQ          ;See if there are any entries on this queue
005572  162705  000002                  SUB     #2,R5           ;Get index number of next CL unit
005576  002365                          BGE     1$              ;Br if more units to check
                                ;
                                ;  Call routine to return any freed queue elements to the system
                                ;
005600  004737  006002'                 CALL    RTNQ            ;Return freed queue elements to the system
                                ;
                                ;  Finished
                                ;
005604  012605                  MOV     (SP)+,R5
005606  012604                  MOV     (SP)+,R4
005610  012603                  MOV     (SP)+,R3
005612  000207                  RETURN
```

```
                                        .SBTTL  CKABTQ -- Check for aborted queue elements
                                ;-----------------------------------------------------------------------
                                ;  CKABTQ is called to check to see if any queue elements belonging to
                                ;  an aborted job are on a specified internal queue.
                                ;  If any queue elements for the aborted job are found, they are placed
                                ;  on the completion queue list.
                                ;
                                ;  Inputs:
                                ;   R3 = Pointer to base of queue head vector for CL units.
                                ;   R4 = # of job being aborted
                                ;   R5 = CL unit index number of queue to check.
                                ;
005614  010246                  CKABTQ: MOV     R2,-(SP)
005616  010346                          MOV     R3,-(SP)
005620  010546                          MOV     R5,-(SP)
                                ;
                                ;  Get address of queue head
                                ;
005622  060305                          ADD     R3,R5           ;Point to queue head for this unit
005624  162705  000000C               SUB     #Q.LINK-Q.BLKN,R5;Make head look like fake queue entry
                                ;
                                ;  Search for entries in the queue
                                ;
005630  010503                  1$:     MOV     R5,R3           ;Point to queue head
005632                                  DISABL                  ;;;** Disable interrupts **
005640  010302                          MOV     R3,R2           ;;;Save address of current entry
005642  016303  000000C         2$:     MOV     Q.LINK-Q.BLKN(R3),R3;;;Get address of next entry
005646  001417                          BEQ     9$              ;;;Br if no entries for job being aborted
005650  120463  000000C               CMPB    R4,Q.JOB-Q.BLKN(R3);;;Is this the job being aborted?
005654  001372                          BNE     2$              ;;;Keep looking if not
                                ;
                                ;  We found an entry for the job being aborted
                                ;  Remove it from our internal queue and place on the completion queue
                                ;
005656  016362  000000C 000000C         MOV     Q.LINK-Q.BLKN(R3),Q.LINK-Q.BLKN(R2);;;Remove from list
005664  013763  000004' 000000C         MOV     CQH,Q.LINK-Q.BLKN(R3);;;Put on completion list
005672  010337  000004'                 MOV     R3,CQH
                                ;
                                ;  Go back and see if there are any more entries to remove
                                ;
005676                                  ENABL                   ;** Enable interrupts **
005704  000751                          BR      1$              ;Go repeat the process
                                ;
                                ;  Finished with this queue
                                ;
005706                          9$:     ENABL                   ;** Enable interrupts **
005714  012605                          MOV     (SP)+,R5
005716  012603                          MOV     (SP)+,R3
005720  012602                          MOV     (SP)+,R2
005722  000207                          RETURN
```

```
                                        .SBTTL  MOVQ    -- Move queue element to internal queue
                                ;-----------------------------------------------------------------------
                                ;  MOVQ is called to move the current queue element
                                ;  onto an internal queue.
                                ;
                                ;  Inputs:
                                ;   R3 = Address of internal queue header
                                ;   R4 = Address of current queue element
                                ;   R5 = CL unit index number
                                ;
005724  010346                  MOVQ:  MOV     R3,-(SP)
005726  010446                         MOV     R4,-(SP)
                                ;
                                ;  Set up R3 to point to queue header but make it look like we are
                                ;  pointing to a queue element.
                                ;
005730  060503                         ADD     R5,R3           ;Point to correct queue head entry
005732  162703  000000C               SUB     #Q.LINK-Q.BLKN,R3;Make it look like pointer to a Q element
                                ;
                                ;  Add queue entry to tail of internal list
                                ;
005736  010400                         MOV     R4,R0           ;Save address of new queue element
005740                                 DISABL                  ;;;** Disable interrupts **
005746  010304                  1$:    MOV     R3,R4           ;;;Remember current queue element address
005750  016303  000000C               MOV     Q.LINK-Q.BLKN(R3),R3;;;Get address of next queue element
005754  001374                         BNE     1$              ;;;Loop till end of list found
005756  010064  000000C               MOV     R0,Q.LINK-Q.BLKN(R4);;;Add new entry to end of list
005762  005060  000000C               CLR     Q.LINK-Q.BLKN(R0);;;Say it is the end of the list
                                ;
                                ;  Finished
                                ;
005766                          9$:    ENABL                   ;** Enable interrupts **
005774  012604                         MOV     (SP)+,R4
005776  012603                         MOV     (SP)+,R3
006000  000207                         RETURN
```

```
                                        .SBTTL  RTNQ    -- Return completed queue elements to the system
                                ;-------------------------------------------------------------------------
                                ;  RTNQ is called to return completed queue elements to the system.
                                ;
                                ;  Inputs:
                                ;   CQH = Pointer to 1st queue element on list of completed queue elements.
                                ;
006002  010446                  RTNQ:   MOV     R4,-(SP)
006004  010546                          MOV     R5,-(SP)
                                ;
                                ;  See if this routine is currently being used by someone else.
                                ;  If so, just exit. The other user will return all pending queue elements.
                                ;
006006  005237  000002'                 INC     RTNCNT          ;Is someone else already in this routine?
006012  001072                          BNE     3$              ;Br if yes -- They will return all entries
                                ;
                                ;  No one else is currently in this routine.
                                ;  See if the handler is currently being held.
                                ;
006014                                  DISABL                  ;** Disable interrupts **
006022  005737  000000G                 TST     CLABF           ;;;Is handler currently being held?
006026  002023                          BGE     6$              ;;;Br if not being held
                                ;
                                ;  Handler is being held.
                                ;  This means an I/O abort is being done for the handler.
                                ;  We cannot return queue elements to the system now.
                                ;  Queue a fork request at a low priority which will be held until the
                                ;  I/O abort operation is completed.
                                ;
006030  005737  000006'                 TST     ABTQFL          ;;;Have we already queued a fork request?
006034  001061                          BNE     3$              ;;;Br if yes
006036  005237  000006'                 INC     ABTQFL          ;;;Set flag saying abort fork request queued
006042                                  ENABL                   ;** Enable interrupts **
006050  004737  000000G                 CALL    FRKGET          ;Get a free fork request block
006054  112764  177770  000000G         MOVB    #<FP$IOA-1>,FQ$PRI(R4);Set priority below I/O abort
006062  012764  006002'  000000G         MOV     #RTNQ,FQ$RTN(R4);Set address of routine to be called by fork
006070  004737  000000G                 CALL    FORKQ           ;Queue the fork request
006074  000441                          BR      3$              ;Exit for now -- Fork will recall us
                                ;
                                ;  This handler is not being held.
                                ;  Remove completed queue element from completion list and place it as
                                ;  the current queue element for this handler.
                                ;
006076  005037  000006'         6$:     CLR     ABTQFL          ;;;Say abort fork request no longer queued
006102  013704  000004'         5$:     MOV     CQH,R4          ;;;Get addr of 1st queue element on compl list
006106  001434                          BEQ     3$              ;;;Br if no more entries to free
006110  016437  000000C 000004'         MOV     Q.LINK-Q.BLKN(R4),CQH ;;;Remove entry from completion list
006116  013746  000000G                 MOV     CLCQE,-(SP)     ;;;Save current queue element pointer
006122  013746  000000G                 MOV     CLLQE,-(SP)     ;;;Also save last queue element pointer
006126  010437  000000G                 MOV     R4,CLCQE        ;;;Set entry being freed as current Q element
006132  010437  000000G                 MOV     R4,CLLQE        ;;;And as last queue element
006136  005064  000000C                 CLR     Q.LINK-Q.BLKN(R4);;;Say this element is only one on list
                                ;
                                ;  Now call the system IOFIN routine to release the queue element
                                ;
006142                          4$:     ENABL                   ;** Enable interrupts **
006150  012704  000000G                 MOV     #CLCQE,R4       ;Point to CQE cell for IOFIN
```

```
      58 006154  004737  0000000                       CALL    IOFIN           ;Free the current queue element
      59                                ;
      60                                ;  Restore saved queue element pointers then go back and see
      61                                ;  if there are more queue elements that need to be freed.
      62                                ;
      63 006160                                 DISABL                          ;;;** Disable interrupts **
      64 006166  012637  0000000                MOV     (SP)+,CLLQE             ;;;Restore saved queue element pointers
      65 006172  012637  0000000                MOV     (SP)+,CLCQE             ;;;
      66 006176  000741                         BR      5$                      ;;;Go back and see if more elements to free
      67                                ;
      68                                ;  There are no more queue entries to be freed
      69                                ;
      70 006200  005337  000002'        3$:     DEC     RTNCNT                  ;;;Say we are exiting this routine
      71 006204                                 ENABL                           ;** Enable interrupts **
      72                                ;
      73                                ;  Finished
      74                                ;
      75 006212  012605                         MOV     (SP)+,R5
      76 006214  012604                         MOV     (SP)+,R4
      77 006216  000207                         RETURN
```

```
                                        .SBTTL  LINON  -- Turn on a communications line
                                ;-----------------------------------------------------------------------
                                ;  LINON is called to turn on a communications line the first time I/O
                                ;  is done to the line.
                                ;
                                ;  Inputs:
                                ;   R5 = CL unit index number.
                                ;
006220                          LINON:
                                ;
                                ;  Set flag saying line is turned on
                                ;
006220  052765  0000000 0000000         BIS     #CM$ON,CL$STA(R5)       ;Say line is turned on
                                ;
                                ;  Assert Data Terminal Ready
                                ;
006226  052765  0000000 0000000         BIS     #CO$DTR,CL$OPT(R5)      ;Say we want DTR on
006234  004737  006242'                CALL    SETDTR                  ;Raise the DTR line
                                ;
                                ;  Finished
                                ;
006240  000207                          RETURN
```

```
                                        .SBTTL  SETDTR -- Set Data Terminal Ready status
                                ;-----------------------------------------------------------------------
                                ;  SETDTR is called to confirm that the Data Terminal Ready status is
                                ;  in agreement with the desired state as specified by the CO$DTR
                                ;  flag in the unit option flag word (CL$OPT(R5)).
                                ;
                                ;  Inputs:
                                ;   R5 = CL unit number index
                                ;
006242  010146                  SETDTR: MOV     R1,-(SP)
006244  010246                          MOV     R2,-(SP)
                                ;
                                ;  See if he wants DTR on or off
                                ;
006246  032765  0000000 0000000         BIT     #CO$DTR,CL$OPT(R5);Is DTR wanted on or off?
006254  001412                          BEQ     1$              ;Br if wanted off
                                ;
                                ;  DTR is wanted on.  See if it is currently on.
                                ;
006256  032765  0000000 0000000         BIT     #CM$DTR,CL$STA(R5)      ;Is DTR currently asserted?
006264  001022                          BNE     9$              ;Br if yes -- all is ok
006266  052765  0000000 0000000         BIS     #CM$DTR,CL$STA(R5)      ;Say we are raising DTR
006274  012700  0000000                 MOV     #MS$DTR,R0              ;Say we want to set DTR
006300  000410                          BR      2$                      ;Go set DTR
                                ;
                                ;  DTR is wanted off.  See if it is currently off.
                                ;
006302  032765  0000000 0000000 1$:     BIT     #CM$DTR,CL$STA(R5)      ;Is DTR currently off?
006310  001410                          BEQ     9$                      ;Br if yes -- all is ok
006312  042765  0000000 0000000         BIC     #CM$DTR,CL$STA(R5)      ;Say we are dropping DTR
006320  005000                          CLR     R0                      ;Say we want to drop DTR
                                ;
                                ;  Call hardware-dependent routine to change DTR status
                                ;
006322  016501  0000000         2$:     MOV     CL$LIX(R5),R1           ;Get line # for this CL unit
006326  004737  0000000                 CALL    SETDSS                  ;Change DTR status
                                ;
                                ;  Finished
                                ;
006332  012602                  9$:     MOV     (SP)+,R2
006334  012601                          MOV     (SP)+,R1
006336  000207                          RETURN
```

```
                                        .SBTTL  SETBRK -- Control break transmission
                                ;-------------------------------------------------------------------
                                ;  SETBRK is called to start or end transmission of a break character
                                ;  to a CL line.
                                ;
                                ;  Inputs:
                                ;   R0 = CM$BRK to start sending break; 0 to stop break.
                                ;   R5 = CL unit index number.
                                ;
006340  010146                  SETBRK: MOV     R1,-(SP)
006342  010246                          MOV     R2,-(SP)
                                ;
                                ;  Call hardware-dependent routine to control break transmission
                                ;
006344  016501  0000000                 MOV     CL$LIX(R5),R1   ;Get line # for this CL unit
006350  016102  0000000                 MOV     LCDTYP(R1),R2   ;Get line control type code
006354                                  OCALL   BRKJMP          ;Call hardware control routine
                                ;
                                ;  Finished
                                ;
006362  012602                          MOV     (SP)+,R2
006364  012601                          MOV     (SP)+,R1
006366  000207                          RETURN
                                ;
                                ;  Dummy routine used as a jump off point to the CDSBRK routine.
                                ;  This is done so that we can use an OCALL to save our overlay number.
                                ;
006370  000172  0000000         BRKJMP: JMP     @CDSBRK(R2)     ;Call hardware routine to control break
      29        000001                  .END
Errors detected:  0

*** Assembler statistics


Work  file  reads:  0
Work  file writes:  0
Size of work file:  216 Words  ( 1 Pages)
Size of core pool:  17920 Words  ( 70 Pages)
Operating  system:  RT-11

Elapsed time:  00:00:33.15
DK:TSCLO,LP:TSCLO=DK:TSCLO.MAC/C/N:SYM
```

Cross reference table (CREF V05.04)

```
CL$XLN    1-32     15-21    21-29    28-29    32-19    32-25*
CLABF     1-47     37-21
CLABRT    1-25     34-9#
CLCLOS    8-5      13-9#
CLCQE     1-48      5-20     5-22*   37-48    37-50*   37-57    37-65*
CLEOFS    1-37      8-283    8-370
CLERR     5-49      6-15     6-19     6-23     6-25     6-37#
CLQSTS    8-90      8-322    9-12#
CLINCP    1-25     15-10#
CLIOQ     1-25      5-13#
CLLQE     1-48      5-24*   37-49    37-51*   37-64*
CLOCPY   21-31     27-19    28-9#
CLPTWD    8-91      8-193    8-244    8-263    8-323    8-329    8-334    8-339    8-344    8-349    8-359    8-364
         11-15#
CLQOK     5-19#     5-77     5-95    10-16
CLQXIT    5-59      6-39     7-10     7-11     8-6      8-36     8-65     8-96     8-119    8-137    8-146    8-155
          8-163     8-171    8-179    8-198    8-207    8-236    8-245    8-264    8-294    8-301    8-383    10-8#
CLREAD    5-58      5-64#    7-34     8-72
CLREST    1-26      8-110    8-300   14-14#   32-14
CLSFAB    1-39
CLSFBC    1-53
CLSFCH    1-53
CLSFCO    1-54
CLSFDL    1-53
CLSFHS    1-53
CLSFIC    1-45
CLSFMS    1-49
CLSFOC    1-45
CLSFRB    1-53     16-76    16-91
CLSFRL    1-50     18-22
CLSFSL    1-54
CLSFSO    1-54
CLSFSP    1-57
CLSFSS    1-54
CLSFSW    1-54
CLSFWB    1-38
CLSPFN    5-55      6-10#
CLSTRT    8-28      8-53     8-61    22-142   28-90    29-31    29-68    30-16    33-8#
CLTIMR    1-25     21-9#
CLTOTL    1-32     21-15    34-15
CLVERS    1-38      9-16
CLWRIT    5-57      5-82#
CLWRTB    5-83#     7-37
CLXBRK    1-25     28-57    31-10#
CLXDRP   29-75     31-19    32-10#
CLXICP    1-25     27-11#
CLXMCC   28-52     29-13#
CLXSSB   29-34     30-8#
CM$BRK    1-57      8-50     8-60    14-46    14-50    29-28    30-15
CM$CRL    1-56     14-42    22-87    26-4     26-22    26-30    26-65    26-67    26-75    26-90
CM$DTR    1-50     39-20    39-22    39-28    39-30
CM$EFP    1-37      8-256    13-18    14-42    22-35    23-18    24-37
CM$EOF    1-44      8-32     8-71    14-42    16-42    16-54    16-58    18-34
CM$FFI    1-34     22-72    22-74    26-60
CM$FFS    1-51     14-42    22-56    22-62    26-53    26-57    26-82
CM$IRG    1-44     16-15    16-22    16-101
```

```
CM$MCC    1-34     28-49     28-51     28-61
CM$ON     1-44      8-106    38-13
CM$ORP    1-48     22-17     22-23     22-147    28-18     28-23     28-95
CM$TBS    1-44     14-42     22-46     22-52     26-40
CM$WRT    1-35      5-83     13-13     13-19     14-42
CO$BBT    1-43     22-82
CO$BNI    1-43     16-74     18-9
CO$BNO    1-43     22-80
CO$CR     1-52     26-92
CO$CTL    1-52     26-5
CO$DTR    1-50      8-114     8-130     8-135     29-47     29-55     31-14     38-17     39-15
CO$FF     1-42     26-76
CO$FFO    1-42     23-37
CO$LC     1-51     22-92
CO$LFI    1-42     18-15
CO$LFO    1-42     26-63
CO$TAB    1-42     26-31
CQ$LNK    1-31     29-38*
CQ$LOT    1-30     29-33*
CQ$PA5    1-30     29-35*
CQ$RO     1-31     29-36*
CQ$RTN    1-30     29-34*
CQH       4-6#     10-9      10-10*    20-31     20-32*    23-61     23-62*    35-36     35-37*    37-45     37-47*
CR        3-5#
CS$EOF    1-50     16-57
CS$ERR    1-41      6-38     11-26     12-28
CTRLQ     3-9#
CTRLS     3-8#
CTRLZ     3-10#
EOFCHR   23-24     24-14#
FF        3-7#     22-75     23-40     24-21     26-59
FORCEX    1-36     32-31
FORKQ     1-46     37-37
FP$IOA    1-46     37-35
FQ$PRI    1-46     37-35*
FQ$RTN    1-46     37-36*
FRKGET    1-46     37-34
GETCHR   22-67     23-14#
GETDSS    1-59      8-189     9-34
GETRTQ    1-30     29-32
GETWRD    8-143     8-152     8-161     8-169     8-177     8-204     8-274     12-17#
GTBYT     1-49      8-284    23-51
INPCHR   16-86     18-4      18-24     19-12#
INTPRI    1-33      5-25     10-11     14-34     16-17     16-23     16-50     16-102    20-33     21-37     22-24     22-42
         22-141    22-151    23-63     28-24     28-39     28-89     28-99     29-40     32-26     35-41     35-46     36-32
         37-33     37-56     37-71
IOFIN     1-34     37-58
IRINGQ    5-71     15-29     16-9#
KPAR5     1-30     29-35
KPAR6     1-52     12-36*
LCDTYP    1-38      1-55      8-23     14-62     29-66     33-16     40-16
LCLUNT    1-58     15-15
LF        3-6#     22-60     26-69     26-83
LHIRBA    1-40      8-242    14-26     16-44     16-93     28-37
LHIRBB    1-40     14-23
LHIRBQ    1-40     14-25*
```

Cross reference table (CREF V05.04)

```
LHIRBP    1-40      14-24*
LHIRBS    1-39       8-243    14-26*    16-44     16-93     28-37
LINIR     1-36      32-21*
LINON     5-64       5-86     38-9#
LNMAP     1-36      32-30
LSW10     1-43       8-22*     9-22     14-59     14-61*    29-65*
LSW3      1-32       1-35      8-18*     8-259     9-28     14-38*    29-64*
LXCL      1-32      27-15     32-20*
MAXSF0    6-16       6-27      7-12#
MAXSF1    6-20       6-26      7-22#
MAXSF2    6-24       7-41#
MOVQ      5-66       5-85     36-11#
MRKTHD    1-31      29-38     29-39*
MS$BRK    1-57       8-51     29-29
MS$CAR    1-56       9-35
MS$DTR    1-55      39-23
MS$RNG    1-56       9-41
NEDCDO    1-39      33-18*
NEDCLO    1-39      33-19*
ORINGP    5-91      13-31     21-33     22-9#
OVRHC     1-55      40-17
PSW       1-33       5-19*     5-25*    10-8*     10-11*    14-22*    14-34*    16-14*    16-17*    16-23*    16-39*    16-50*
         16-102*    20-29*    20-33*    21-37*    22-16*    22-24*    22-30*    22-42*    22-141*   22-151*   23-59*    23-63*
         28-17*     28-24*    28-34*    28-39*    28-89*    28-99*    29-37*    29-40*    32-18*    32-26*    35-25*    35-41*
         35-46*     36-23*    36-32*    37-20*    37-33*    37-56*    37-63*    37-71*
PTBYT     1-38       8-374     8-377    19-22     20-23
PTWRD     1-33      11-34
Q.BLKN    1-34       5-22      5-41      5-43      5-54      5-56*     5-82*     6-37      8-45      8-125     8-217     10-9*
         11-19      11-24     12-21     12-26     12-36     12-40     12-41*    16-56     16-76     16-91     18-22     19-26*
         20-20      20-30     20-31*    23-35     23-35     23-39*    23-39*    23-45     23-50*    23-60     23-61*    35-20
         35-27      35-29     35-35     35-35*    35-36*    36-18     36-25     36-27*    36-28*    37-47     37-52*
Q.BUFF    1-52      11-19     12-21     12-40     12-41*
Q.CSW     1-41       6-37     11-24     12-26     16-56
Q.DEVX    1-30       5-43
Q.FUNC    1-41       5-54     16-76     16-91     18-22
Q.JOB     1-32       8-217    35-29
Q.LINK    1-34       5-22     10-9*     20-30     20-31*    23-60     23-61*    35-20     35-27     35-35     35-35*    35-36*
          36-18     36-25     36-27*    36-28*    37-47     37-52*
Q.PAR     1-52      12-36
Q.UNIT    1-41       5-41
Q.WCNT    1-34       5-56*     5-82*     8-45      8-125     19-26*    20-20     23-45     23-50*
RDFIN    16-59      16-95     18-25     18-38     19-32     20-10#
RTNCNT    4-5#      37-14*    37-70*
RTNQ      8-231     10-12     20-37     23-67     34-25     37-8#     37-36
SETBRK    8-52       8-59     14-49     29-30     30-14     40-10#
SETDSS    1-59      39-36
SETDTR    1-26       8-115     8-136     8-145     8-154     21-24     29-48     29-56     31-15     38-18     39-10#
SETSPD    1-35       8-206
SFABT     7-33       8-216#
SFBREK    7-17       8-45#
SFCLER    7-16       8-14#
SFCLOS    7-8        8-5#
SFCOPT    7-27       8-152#
SFDTR     7-21       8-125#
SFGMS     7-31       8-185#
SFGOPT    7-40       8-318#
```

```
SFGRP0    7-8#       7-12
SFGRP1    7-16#      7-22
SFGRP2    7-26#      7-41
SFIC      7-35       8-242#
SFOC      7-36       8-251#
SFREAD    7-18       8-71#
SFREST    7-39       8-300#
SFSEFP    7-38       8-270#
SFSLEN    7-28       8-161#
SFSOPT    7-26       8-143#
SFSPD     7-32       8-204#
SFSSKP    7-29       8-169#
SFSTAT    7-19       8-88#
SFSWID    7-30       8-177#
SFTERM    7-9        7-20       8-102#
SILFET    1-45      16-64      28-43
SPACE     3-11#     22-50      26-42
SPFRTN    6-33       7-4#
TRNSTR    1-33      15-23
TTINCP    1-36      32-21
VCXCTL    1-31      28-59
VCXTRM    1-31      28-55
XL$CD     1-59       9-37
XL$CTS    1-59       9-37
XL$RI     1-59       9-43
XL$XFR    1-59       9-30
XL$XFX    1-59       9-24
```

Cross reference table (CREF V05.04)

DISABL	2-7#	5-19	10-8	14-22	16-14	16-39	20-29	22-16	22-30	23-59	28-17	28-34
	29-37	32-18	35-25	36-23	37-20	37-63						
ENABL	2-13#	5-25	10-11	14-34	16-17	16-23	16-50	16-102	20-33	21-37	22-24	22-42
	22-141	22-151	23-63	28-24	28-39	28-89	28-99	29-40	32-26	35-41	35-46	36-32
	37-33	37-56	37-71									
OCALL	2-19#	40-17										