TSCLO -- Communication Line (CL MACRO V05.05 Wednesday 18-Jan-89 15:23
Table of contents

```
                                        .TITLE  TSCLO -- Communication Line (CL) Handler for TSX-Plus
                                        .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,CLCLER
                                ;
                                ;  Global references
                                ;
                                        .GLOBL  GETRTQ,CQ$LOT,CQ$RTN,KPAR5,CQ$PA5,Q.DEVX
                                        .GLOBL  CQ$RO,MRKTHD,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,$HISTP,CLSFAB,NEDCDO,NEDCLO
                                        .GLOBL  LHIRBB,LHIRBG,LHIRBP,LHIRBA
                                        .GLOBL  Q.UNIT,Q.FUNC,Q.CSW,CS$ERR,C.CSW
                                        .GLOBL  CO$FF,CO$TAB,CO$LFO,CO$LFI,CO$FFO
                                        .GLOBL  CO$BNO,CO$BNI,LSW10,CO$8BT
                                        .GLOBL  CM$TBS,CM$IRG,CM$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,GTBYT,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,CDGDSS,OVRHC,LCDTYP
                                        .GLOBL  CM$CRL,CDGDSS,MS$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  000000G                 MOV     CLCQE,R4        ;;;Get pointer to queue element
000030  001406                          BEQ     1$              ;;;Br if there is no queue element to process
000032  016437  000000C 000000G         MOV     Q.LINK-Q.BLKN(R4),CLCQE;;;Remove queue element from list
000040  001002                          BNE     1$              ;;;Br if more elements pending
000042  005037  000000G                 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$:     MOVB    Q.UNIT-Q.BLKN(R4),R5;Get device unit number
000074  042705  177770                  BIC     #^C7,R5         ;Clear all but unit # field
000100  126437  000000C 000000G         CMPB    Q.DEVX-Q.BLKN(R4),C1DEVX ;Is the a C1 unit?
000106  001002                          BNE     4$              ;Br if not
000110  062705  000010                  ADD     #8.,R5          ;Bias C1 unit numbers by 8
000114  006305                  4$:     ASL     R5              ;Convert to word index
000116  016501  000000G                 MOV     CL$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
```

```
     58 000144  001002                          BNE     CLREAD          ;Br if this is a read operation
     59 000146  000137  001502'                  JMP     CLQXIT          ;Br if this is a seek operation
     60                                 ;
     61                                 ;  This is a .READ operation.
     62                                 ;  Move queue entry to internal read queue for this unit.
     63                                 ;
     64 000152  004737  006226'         CLREAD: CALL    LINON           ;Turn on the line
     65 000156  012703  000000G                 MOV     #CL$RQH,R3      ;Get pointer to read queue head
     66 000162  004737  005732'                 CALL    MOVQ            ;Move queue element to internal queue
     67                                 ;
     68                                 ;  Call routine to move any pending characters in silo buffer for this
     69                                 ;  line into the data buffer.
     70                                 ;
     71 000166  004737  002212'                 CALL    IRINGQ          ;Move chars from silo buffer to data buffer
     72                                 ;
     73                                 ;  Finished starting the read operation.
     74                                 ;  The queue element will be returned to the system when we have
     75                                 ;  completed the operation.
     76                                 ;
     77 000172  000137  000016'                 JMP     CLQOK
     78                                 ;
     79                                 ;  This is a .WRITE request.
     80                                 ;  Move queue element to internal write queue for this CL unit.
     81                                 ;
     82 000176  005464  000000C         CLWRIT: NEG     Q.WCNT-Q.BLKN(R4) ;Make write byte count positive
     83 000202  052765  000000G 000000G CLWRTB: BIS     #CM$WRT,CL$STA(R5);Set flag that says a write has been done
     84 000210  012703  000000G                 MOV     #CL$WQH,R3      ;Get pointer to write queue head
     85 000214  004737  005732'                 CALL    MOVQ            ;Move queue element to write queue
     86 000220  004737  006226'                 CALL    LINON           ;Turn on the line
     87                                 ;
     88                                 ;  Move characters from data buffer to output ring buffer and then
     89                                 ;  start output to the line.
     90                                 ;
     91 000224  004737  003130'                 CALL    ORINGP          ;Move chars from data buffer to ring buffer
     92                                 ;
     93                                 ;  Finished starting a write operation
     94                                 ;
     95 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  0000000 0000000         BIS     #CS$ERR,C.CSW(R0) ;Set hard error flag in CSW
000334  000137  001502'                 JMP     CLQXIT          ;Do .DRFIN to tell system this op is completed
```

```
                                ;-----------------------------------------------------------------------
                                ;  Branch vector for .SPFUN processing routines based on function code value.
                                ;
000340                          SPFRTN:
                                ;
                                ;  Group 0: Function codes in the range 1 to 4
                                ;
000340  000422'                 SFGRP0: .WORD   SFCLOS          ;001 - Close file
000342  000552'                         .WORD   SFTERM          ;002 - Delete file
000344  001502'                         .WORD   CLQXIT          ;003 - Lookup file
000346  001502'                         .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  000442'                         .WORD   SFBREK          ;202 - Break transmission control
000354  000520'                         .WORD   SFREAD          ;203 - Special read with byte count
000356  000532'                         .WORD   SFSTAT          ;204 - Get handler status
000360  000552'                         .WORD   SFTERM          ;205 - Terminate I/O
000362  000602'                         .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  000636'                 SFGRP2: .WORD   SFSOPT          ;250 - Set option flags
000366  000656'                         .WORD   SFCOPT          ;251 - Clear option flags
000370  000676'                         .WORD   SFSLEN          ;252 - Set page length
000372  000712'                         .WORD   SFSSKP          ;253 - Set skip lines
000374  000726'                         .WORD   SFSWID          ;254 - Set page width
000376  000742'                         .WORD   SFGMS           ;255 - Get modem status
000400  000762'                         .WORD   SFSPD           ;256 - Set transmit/receive speed
000402  001000'                         .WORD   SFABT           ;257 - Abort all pending read/write requests
000404  000152'                         .WORD   CLREAD          ;260 - Read line with byte count
000406  001040'                         .WORD   SFIC            ;261 - Get number of pending input characters
000410  001060'                         .WORD   SFOC            ;262 - Get number of pending output chars
000412  000202'                         .WORD   CLWRTB          ;263 - Write with byte count
000414  001136'                         .WORD   SFSEFP          ;264 - Set end-of-file output control
000416  001214'                         .WORD   SFREST          ;265 - Reset CL unit
000420  001224'                         .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  001660'        SFCLOS: CALL    CLCLOS          ;Perform end-of-file operations
000426  000137  001502'                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:
                                ;
                                ;  R1 contains index of TS line in use as CL unit
                                ;  R5 contains CL unit index
                                ;  Call the routine shared with the equivalent EMT to do the work
                                ;
000432  004737  001724'                CALL    CLCLER          ;Go do it
000436  000137  001502'                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.
                                ;
                                ;
000442  005764  000000C         SFBREK: TST     Q.WCNT-Q.BLKN(R4) ;Start or end break?
000446  001412                        BEQ     1$              ;Br if we are ending a break
                                ;
                                ;  Begin sending a break
                                ;
000450  052765  0000000 0000000         BIS     #CM$BRK,CL$STA(R5);Set flag saying we are sending a break
000456  012700  0000000                 MOV     #MS$BRK,R0      ;Set flag to start break transmission
000462  004737  006346'                 CALL    SETBRK          ;Call hardware routine to start sending break
000466  004737  005514'                 CALL    CLSTRT          ;Start transmitter
000472  000410                         BR      9$
                                ;
                                ;  End sending a break
                                ;
000474  005000                  1$:     CLR     R0              ;Clear break-send flag
000476  004737  006346'                 CALL    SETBRK          ;Call hardware routine to end break
000502  042765  0000000 0000000         BIC     #CM$BRK,CL$STA(R5);Clear flag that says we are sending a break
000510  004737  005514'                 CALL    CLSTRT          ;Start transmitter
                                ;
                                ;  Finished
                                ;
000514  000137  001502'         9$:     JMP     CLQXIT          ;Finished with .SPFUN
                                ;
                                ;-------------------------------------------------------------------------
                                ;  Special function # 203
                                ;  Read with Q.WCNT indicating the byte count rather than the word count
                                ;
000520  042765  0000000 0000000 SFREAD: BIC     #CM$EOF,CL$STA(R5);Clear end of file status
000526  000137  000152'                 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$RI  bit 4:  1 ==> Ring is detected
                                        ;
000532  010246                          SFSTAT: MOV     R2,-(SP)
                                        ;
000534  004737  001400'                         CALL    CLGSTS          ;Call common routine to get status
000540  004737  001540'                         CALL    CLPTWD          ;Store value into user's buffer
                                        ;
                                        ;  Finished
                                        ;
000544  012602                                  MOV     (SP)+,R2
000546  000137  001502'                         JMP     CLQXIT          ;Finished with operation
                                        ;
                                        ;-------------------------------------------------------------------------------
                                        ;  Special function # 205
                                        ;  Terminate I/O to the line.
                                        ;
000552                                  SFTERM:
                                        ;
                                        ;  Set flag saying to ignore input from the line
                                        ;
000552  042765  000000G 000000G                 BIC     #CM$ON,CL$STA(R5);Say line is turned off
                                        ;
                                        ;  Clear input and output silos and other CL unit status
                                        ;
000560  004737  001764'                         CALL    CLREST          ;Reset the CL unit
                                        ;
                                        ;  Drop Data Terminal Ready
                                        ;
000564  042765  000000G 000000G                 BIC     #CO$DTR,CL$OPT(R5) ;Say we want DTR off
000572  004737  006250'                         CALL    SETDTR          ;Call routine to drop DTR
                                        ;
                                        ;  Finished
                                        ;
000576  000137  001502'                         JMP     CLQXIT
                                        ;
                                        ;-------------------------------------------------------------------------------
                                        ;  Special function # 206
                                        ;  Raise or drop DTR signal.
                                        ;
000602  005764  000000C                 SFDTR:  TST     Q.WCNT-Q.BLKN(R4) ;Raise or drop DTR?
000606  001004                                  BNE     1$              ;Br if raising DTR
                                        ;
                                        ;  Drop DTR
                                        ;
```

```
     115 000610  042765  000000G 000000G          BIC     #CO$DTR,CL$OPT(R5) ;Drop DTR
     116 000616  000403                           BR      2$
     117                                ;
     118                                ;  Raise DTR
     119                                ;
     120 000620  052765  000000G 000000G 1$:      BIS     #CO$DTR,CL$OPT(R5) ;Raise DTR
     121 000626  004737  006250'          2$:      CALL    SETDTR             ;Call routine to raise or drop DTR
     122 000632  000137  001502'                   JMP     CLQXIT             ;Finished with .SPFUN
     123                                ;
     124                                ;-------------------------------------------------------------------------------
     125                                ;  Special function # 250
     126                                ;  Set option flags
     127                                ;
     128 000636  004737  001602'          SFSOPT:  CALL    GETWRD             ;Get word from user's buffer
     129 000642  050065  000000G                   BIS     R0,CL$OPT(R5)      ;Set specified option flags
     130 000646  004737  006250'                   CALL    SETDTR             ;Check for DTR status change
     131 000652  000137  001502'                   JMP     CLQXIT
     132                                ;
     133                                ;-------------------------------------------------------------------------------
     134                                ;  Special function # 251
     135                                ;  Clear option flags
     136                                ;
     137 000656  004737  001602'          SFCOPT:  CALL    GETWRD             ;Get word from user's buffer
     138 000662  040065  000000G                   BIC     R0,CL$OPT(R5)      ;Clear specified option flags
     139 000666  004737  006250'                   CALL    SETDTR             ;Check for DTR status change
     140 000672  000137  001502'                   JMP     CLQXIT
     141                                ;
     142                                ;-------------------------------------------------------------------------------
     143                                ;  Special function # 252
     144                                ;  Set page length
     145                                ;
     146 000676  004737  001602'          SFSLEN:  CALL    GETWRD             ;Get word from user's buffer
     147 000702  010065  000000G                   MOV     R0,CL$LEN(R5)      ;Set page length for this unit
     148 000706  000137  001502'                   JMP     CLQXIT
     149                                ;
     150                                ;-------------------------------------------------------------------------------
     151                                ;  Special function # 253
     152                                ;  Set number of lines to skip at bottom of page.
     153                                ;
     154 000712  004737  001602'          SFSSKP:  CALL    GETWRD             ;Get word from user's buffer
     155 000716  010065  000000G                   MOV     R0,CL$SKP(R5)      ;Set skip lines
     156 000722  000137  001502'                   JMP     CLQXIT
     157                                ;
     158                                ;-------------------------------------------------------------------------------
     159                                ;  Special function # 254
     160                                ;  Set line width.
     161                                ;
     162 000726  004737  001602'          SFSWID:  CALL    GETWRD             ;Get word from user's buffer
     163 000732  010065  000000G                   MOV     R0,CL$WID(R5)      ;Set line width
     164 000736  000137  001502'                   JMP     CLQXIT
     165                                ;
     166                                ;-------------------------------------------------------------------------------
     167                                ;  Special function # 255
     168                                ;  Get modem status
     169                                ;
     170 000742  010246                 SFGMS:   MOV     R2,-(SP)
     171                                ;
```

```
                                        ;  Call hardware dependent routine to get the modem status
                                        ;
000744  004737  000000G                       CALL    GETDSS          ;Call routine to get the data set status
                                        ;
                                        ;  Return status value to 1st word of user's buffer
                                        ;
000750  004737  001540'                       CALL    CLPTWD          ;Store value into 1st word of user's buffer
                                        ;
                                        ;  Finished
                                        ;
000754  012602                                MOV     (SP)+,R2
000756  000137  001502'                       JMP     CLQXIT          ;Finished I/O operation
                                        ;
                                        ;-------------------------------------------------------------------------
                                        ;  Special function # 256.
                                        ;  Set transmit/receive speed.
                                        ;
000762  004737  001602'         SFSPD:  CALL    GETWRD          ;Get word from user's buffer
000766  103402                         BCS     1$              ;Br if invalid buffer address
000770  004737  000000G                CALL    SETSPD          ;Set the speed
000774  000137  001502'         1$:     JMP     CLQXIT          ;Finished
                                        ;
                                        ;-------------------------------------------------------------------------
                                        ;  Special function # 257.
                                        ;  Abort all pending read and write requests for the job.
                                        ;
                                        ;  Inputs:
                                        ;   R4 = Pointer to 3rd word of .SPFUN queue element.
                                        ;
001000  010446                  SFABT:  MOV     R4,-(SP)        ;Save pointer to current queue element
001002  116404  000000C                MOVB    Q.JOB-Q.BLKN(R4),R4 ;Get job # from .SPFUN queue element
                                        ;
                                        ;  Abort pending read requests for this job
                                        ;
001006  012703  000000G                MOV     #CL$RQH,R3      ;Point to read queue head
001012  004737  005622'                CALL    CKABTQ          ;Abort pending reads for job
                                        ;
                                        ;  Abort pending write requests for this job
                                        ;
001016  012703  000000G                MOV     #CL$WQH,R3      ;Point to write queue head
001022  004737  005622'                CALL    CKABTQ          ;Abort pending writs for job
                                        ;
                                        ;  Call routine to return any freed queue elements to the system
                                        ;
001026  004737  006010'                CALL    RTNQ            ;Return freed queue elements to the system
                                        ;
                                        ;  Finished
                                        ;
001032  012604                         MOV     (SP)+,R4        ;Restore pointer to queue element
001034  000137  001502'                JMP     CLQXIT          ;Finished operation
                                        ;
                                        ;-------------------------------------------------------------------------
                                        ;  Special function # 261.
                                        ;  Get number of bytes pending in input silo buffer.
                                        ;
001040  016100  000000G         SFIC:   MOV     LHIRBA(R1),R0   ;Get allocated size of input buffer
001044  166100  000000G                SUB     LHIRBS(R1),R0   ;Subtract free space to get # chars in buf
```

```
    229 001050  004737  001540'                 CALL    CLPTWD          ;Store value into user's buffer
    230 001054  000137  001502'                 JMP     CLQXIT          ;Finished with operation
    231                                 ;
    232                                 ;-----------------------------------------------------------------
    233                                 ;  Special function # 262.
    234                                 ;  Get number of bytes pending in output ring buffer.
    235                                 ;
    236 001060  010446                  SFOC:   MOV     R4,-(SP)
    237 001062  016500  000000G                 MOV     CL$ORA(R5),R0   ;Get allocated space for output ring buffer
    238 001066  166500  000000G                 SUB     CL$ORS(R5),R0   ;Subtract free space to get # chars in buf
    239 001072  016504  000000G                 MOV     CL$LIX(R5),R4   ;Get index # of line we are assigned to
    240 001076  001412                          BEQ     1$              ;Br if not assigned to a line
    241 001100  032765  000000G 000000G        BIT     #CM$EFP,CL$STA(R5);Are we doing end-of-file processing?
    242 001106  001401                          BEQ     2$              ;Br if not
    243 001110  005200                          INC     R0              ;Add an extra character
    244 001112  032764  000000G 000000G 2$:    BIT     #$XCHAR,LSW3(R4);Is output transmission going to line now?
    245 001120  001401                          BEQ     1$              ;Br if not
    246 001122  005200                          INC     R0              ;Say another character pending for output
    247 001124  012604                  1$:     MOV     (SP)+,R4        ;Restore pointer into queue element
    248 001126  004737  001540'                 CALL    CLPTWD          ;Store value into user's buffer
    249 001132  000137  001502'                 JMP     CLQXIT          ;Finished with operation
    250                                 ;
    251                                 ;-----------------------------------------------------------------
    252                                 ;  Special function # 264.
    253                                 ;  Set end-of-file output processing control information.
    254                                 ;
    255 001136                          SFSEFP:
    256                                 ;
    257                                 ;  Set form-feed count
    258                                 ;
    259 001136  004737  001602'                 CALL    GETWRD          ;Get form-feed count from user's buffer
    260 001142  103422                          BCS     9$              ;Br if invalid buffer address
    261 001144  120027  000377                  CMPB    R0,#377         ;Don't change form-feed count?
    262 001150  001402                          BEQ     2$              ;Br if don't-change value
    263 001152  010065  000000G                 MOV     R0,CL$EPN(R5)   ;Set # form-feeds to send at end-of-file
    264                                 ;
    265                                 ;  Set up end-of-file output string
    266                                 ;
    267 001156  016502  000000G         2$:     MOV     CL$EPS(R5),R2   ;Get pointer to area where string is stored
    268 001162  012703  000000G                 MOV     #CLEOFS,R3      ;Get max # bytes allowed for string
    269 001166  004737  000000G         1$:     CALL    GTBYT           ;Get next byte from string
    270 001172  121627  000377                  CMPB    (SP),#377       ;Don't change string?
    271 001176  001404                          BEQ     9$              ;Br if don't change
    272 001200  112622                          MOVB    (SP)+,(R2)+     ;Move char to string area
    273 001202  001402                          BEQ     9$              ;Br if this is end of string
    274 001204  077310                          SOB     R3,1$           ;Loop if we can get more chars
    275 001206  105022                          CLRB    (R2)+           ;Terminate string with null
    276                                 ;
    277                                 ;  Finished
    278                                 ;
    279 001210  000137  001502'         9$:     JMP     CLQXIT          ;Finished
    280                                 ;
    281                                 ;-----------------------------------------------------------------
    282                                 ;  Special function # 265.
    283                                 ;  Reset CL unit.
    284                                 ;
    285 001214  004737  001764'         SFREST: CALL    CLREST          ;Call routine to reset CL unit status
```

```
001220  000137  001502'                JMP     CLQXIT          ;Finished
                                ;
                                ;-------------------------------------------------------------------------
                                ;  Special function # 266
                                ;  Get current options and settings.
                                ;
                                ;  Returns 13 words to user buffer
                                ;  1    Handler status as for SPFUN 204
                                ;  2    CL options flags
                                ;  3    internal flags word
                                ;  4    page length
                                ;  5    end of page skip lines
                                ;  6    page width
                                ;  7    TS line number and CL unit number
                                ;  8    number end of file form feeds
                                ;  9-12 end of file string, ASCIZ up to CLEOFS long
                                ;
001224                          SFGOPT:
                                ;
                                ;  Get and return status word just as for SPFUN 204
                                ;  word 1
001224  004737  001400'                 CALL    CLGSTS          ;Get current status word
001230  004737  001540'                 CALL    CLPTWD          ;Return to user buffer
001234  103457                         BCS     9$              ;Error return if no chan or odd buff addr
                                ;
                                ;  Return CL options word
                                ;  word 2
001236  016500  000000G                 MOV     CL$OPT(R5),R0   ;Get options word
001242  004737  001540'                 CALL    CLPTWD          ;Return to user buffer
                                ;
                                ;  Return internal status word
                                ;  word 3
001246  016500  000000G                 MOV     CL$STA(R5),R0   ;Get internal status word
001252  004737  001540'                 CALL    CLPTWD          ;Return to user buffer
                                ;
                                ;  Return current page length
                                ;  word 4
001256  016500  000000G                 MOV     CL$LEN(R5),R0   ;Get current length
001262  004737  001540'                 CALL    CLPTWD          ;Return to user buffer
                                ;
                                ;  Return current number lines to skip at end of page
                                ;  word 5
001266  016500  000000G                 MOV     CL$SKP(R5),R0   ;Get current # skip lines
001272  004737  001540'                 CALL    CLPTWD          ;Return to user buffer
                                ;
                                ;  Return current page width
                                ;  word 6
001276  016500  000000G                 MOV     CL$WID(R5),R0   ;Get current width
001302  004737  001540'                 CALL    CLPTWD          ;Return to user buffer
                                ;
                                ;  Line number being used as CL unit --> low byte
                                ;  CL unit number --> high byte (will be > 7 if C1 unit);
                                ;  word 7
001306  010146                          MOV     R1,-(SP)        ;Get T/S line index
001310  010500                          MOV     R5,R0           ;Get CL unit index
001312  000300                          SWAB    R0              ;Move to high byte
001314  052600                          BIS     (SP)+,R0        ;Merge in line index
```

```
    343 001316  006200                          ASR     R0              ;Convert indices to numbers
    344 001320  004737  001540'                 CALL    CLPTWD          ;Return to user buffer
    345                                 ;
    346                                 ;  Return current # end of file form feeds
    347                                 ;  word 8
    348 001324  016500  000000G                 MOV     CL$EPN(R5),R0   ;Get EOF FF's
    349 001330  004737  001540'                 CALL    CLPTWD          ;Return to user buffer
    350                                 ;
    351                                 ;  Return current end of file string
    352                                 ;  words 9 - 12
    353 001334  010246                          MOV     R2,-(SP)        ;Save registers
    354 001336  010346                          MOV     R3,-(SP)
    355 001340  012703  000000G                 MOV     #CLEOFS,R3      ;Get number of chars to move
    356 001344  016502  000000G                 MOV     CL$EPS(R5),R2   ;Get pointer to EOF string
    357 001350  112246                  1$:     MOVB    (R2)+,-(SP)     ;Get next character
    358 001352  001404                          BEQ     2$              ;Stop at end of string
    359 001354  004737  000000G                 CALL    PTBYT           ;Else move to user buffer
    360 001360  077305                          SOB     R3,1$           ;Move up to maximum length
    361 001362  005046                          CLR     -(SP)           ;Always return ASCIZ string
    362 001364  004737  000000G         2$:     CALL    PTBYT           ;Move last char to user buffer
    363 001370  012603                          MOV     (SP)+,R3        ;Restore registers
    364 001372  012602                          MOV     (SP)+,R2
    365                                 ;
    366                                 ;  Finished
    367                                 ;
    368 001374  000137  001502'         9$:     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)
                                ;
001400  010346                  CLGSTS: MOV     R3,-(SP)        ;Save R3
                                ;
                                ;  Get version number to high-order byte
                                ;
001402  113703  000000G                 MOVB    CLVERS,R3       ;Get version number
001406  042703  177400                  BIC     #^C377,R3       ;Kill possible sign extension
001412  000303                          SWAB    R3              ;Move version to high byte
                                ;
                                ;  See if we have sent an XOFF to stop transmission to us
                                ;
001414  032761  000000G 000000G         BIT     #$HISTP,LSW10(R1);Have we send XOFF?
001422  001402                          BEQ     1$              ;Br if not
001424  052703  000000G                 BIS     #XL$XFX,R3      ;Set status flag
                                ;
                                ;  See if we have received an XOFF
                                ;
001430  032761  000000G 000000G 1$:     BIT     #$CTRLS,LSW3(R1);Have we received an XOFF?
001436  001402                          BEQ     2$              ;Br if not
001440  052703  000000G                 BIS     #XL$XFR,R3      ;Set status flag
                                ;
                                ;  See if Clear To Send (CTS) is asserted
                                ;
001444  004737  000000G         2$:     CALL    GETDSS          ;Call routine to get dataset status
001450  032700  000000G                 BIT     #MS$CAR,R0      ;Is carrier detected?
001454  001402                          BEQ     3$              ;Br if not
001456  052703  000000C                 BIS     #<XL$CTS!XL$CD>,R3 ;Say CTS is asserted and ring detected
                                ;
                                ;  See if Ring is asserted
                                ;
001462  032700  000000G         3$:     BIT     #MS$RNG,R0      ;Is ring detected?
001466  001402                          BEQ     4$              ;Br if not
001470  052703  000000G                 BIS     #XL$RI,R3       ;Say ring is detected
                                ;
                                ;  Return status value in R0
                                ;
001474  010300                  4$:     MOV     R3,R0           ;Get value to R0 for CLPTWD
001476  012603                          MOV     (SP)+,R3        ;Restore R3
001500  000207                          RETURN
```

```
                                        ;-------------------------------------------------------------------------------
                                        ;  We completed the I/O operation.
                                        ;  Return the queue element to the system.
                                        ;
                                        ;  Inputs:
                                        ;   R4 = Address of current queue element.
                                        ;
001502                                  CLQXIT: DISABL                  ;;;** Disable interrupts **
001510  013764  000004' 000000C                 MOV     CQH,Q.LINK-Q.BLKN(R4);;;Put queue element on completed list
001516  010437  000004'                         MOV     R4,CQH          ;;;
001522                                          ENABL                   ;** Enable interrupts
001530  004737  006010'                         CALL    RTNQ            ;Return queue element to the system
                                        ;
                                        ;  Go back and see if there is another queue element pending
                                        ;
001534  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
                                ;
001540                          CLPTWD:
                                ;
                                ;  See if the buffer address is odd
                                ;
001540  032764  000001  000000C         BIT     #1,Q.BUFF-Q.BLKN(R4) ;Is the buffer address odd?
001546  001410                          BEQ     1$                  ;Br if not
                                ;
                                ;  Error: The buffer address is odd
                                ;
001550  016400  000000C                 MOV     Q.CSW-Q.BLKN(R4),R0 ;Get address of CSW for channel
001554  001403                          BEQ     2$                  ;Br if no channel address
001556  052760  000000G 000000G         BIS     #CS$ERR,C.CSW(R0);Set error flag in CSW
001564  000261                  2$:     SEC                         ;Signal error on return
001566  000404                          BR      9$
                                ;
                                ;  Buffer address is OK.
                                ;  Call PTWRD to store the value.
                                ;
001570  010046                  1$:     MOV     R0,-(SP)            ;Stack the value for PTWRD
001572  004737  000000G                 CALL    PTWRD               ;Store value into user's buffer
001576  000241                          CLC                         ;Signal success on return
                                ;
                                ;  Finished
                                ;
001600  000207                  9$:     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.
                                ;
001602                          GETWRD:
                                ;
                                ;  See if the buffer address is odd
                                ;
001602  032764  000001  000000C         BIT     #1,Q.BUFF-Q.BLKN(R4)    ;Is the buffer address odd?
001610  001411                         BEQ     1$                      ;Br if not odd
                                ;
                                ;  Error: The buffer address is odd
                                ;
001612  016400  000000C                 MOV     Q.CSW-Q.BLKN(R4),R0     ;Get address of channel status word
001616  001403                         BEQ     2$                      ;Br if there is none
001620  052760  000000G 000000G         BIS     #CS$ERR,C.CSW(R0)       ;Set error flag in channel status
001626  005000                 2$:     CLR     R0                      ;Return 0 in R0
001630  000261                         SEC                             ;Signal error on return
001632  000411                         BR      9$
                                ;
                                ;  Buffer address is ok.
                                ;  Map PAR6 to user's buffer.
                                ;
001634  016437  000000C 000000G 1$:     MOV     Q.PAR-Q.BLKN(R4),@#KPAR6 ;Map KPAR6 to user's buffer
                                ;
                                ;  Get word from the buffer
                                ;
001642  017400  000000C                 MOV     @Q.BUFF-Q.BLKN(R4),R0   ;Get value from buffer
001646  062764  000002  000000C         ADD     #2,Q.BUFF-Q.BLKN(R4)    ;Advance buffer address
001654  000241                         CLC                             ;Signal success on return
                                ;
                                ;  Finished
                                ;
001656  000207                 9$:     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
                                ;
001660                          CLCLOS:
                                ;
                                ;  Only do output EOF processing if a write was done to this unit
                                ;
001660  032765  000000G 000000G         BIT     #CM$WRT,CL$STA(R5);Was a write done to this unit?
001666  001415                         BEQ     9$                 ;Br if not
                                ;
                                ;  Say we are doing end-of-file processing
                                ;
001670  052765  000000G 000000G         BIS     #CM$EFP,CL$STA(R5);We have started EOF processing for unit
001676  042765  000000G 000000G         BIC     #CM$WRT,CL$STA(R5);Clear write-done flag for unit
                                ;
                                ;  Reset form-feed count
                                ;
001704  105065  000001G                 CLRB    CL$EPN+1(R5)    ;Say no form-feeds sent yet
                                ;
                                ;  Reset ENDSTRING pointer
                                ;
001710  016565  000000G 000000G         MOV     CL$EPS(R5),CL$EPP(R5);Reset endstring pointer
                                ;
                                ;  Initiate output to the unit
                                ;
001716  004737  003130'                 CALL    ORINGP          ;Initiate output to unit
                                ;
                                ;  Finished
                                ;
001722  000207                 9$:     RETURN
```

```
                                        .SBTTL  CLCLER -- Clear CL XOFF status
                                ;-----------------------------------------------------------------
                                ;  This routine clears the flag saying that the line has received an
                                ;  XOFF and transmits an XON
                                ;
                                ;  Inputs:
                                ;       R1      Index of the TS line to which the CL unit is connected
                                ;       R5      CL unit index
                                ;
001724                          CLCLER:
                                ;
                                ;  Clear flag saying we have received an XOFF
                                ;
001724  042761  000000G 000000G         BIC     #$CTRLS,LSW3(R1);Clear the ctrl-S flag
                                ;
                                ;  Send an XON
                                ;
001732  042761  000000G 000000G         BIC     #$HISTP,LSW10(R1);Say input has not been stopped by XOFF
001740  016100  000000G                 MOV     LCDTYP(R1),R0   ;Get device type code
001744  004770  000000G                 CALL    @CDSXON(R0)     ;Call routine to stuff XON into output
                                ;
                                ;  Start output
                                ;
001750  004737  005514'                 CALL    CLSTRT          ;Start transmission
                                ;
                                ;  Clear end of file flag
                                ;
001754  042765  000000G 000000G         BIC     #CM$EOF,CL$STA(R5);Clear end of file status
001762  000207                  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
                                ;
001764  010146                  CLREST: MOV     R1,-(SP)
                                ;
                                ;  Get line # CL unit is connected to
                                ;
001766  016501  000000G                 MOV     CL$LIX(R5),R1   ;Get line index number
                                ;
                                ;  Clear out the input silo buffer
                                ;
001772                                  DISABL                  ;;;** Disable interrupts **
002000  016100  000000G                 MOV     LHIRBB(R1),R0   ;;;Get pointer to start of silo buffer
002004  010061  000000G                 MOV     R0,LHIRBP(R1)   ;;;Reset input interrupt pointer
002010  010061  000000G                 MOV     R0,LHIRBG(R1)   ;;;Reset next available char pointer
002014  016161  000000G 000000G         MOV     LHIRBA(R1),LHIRBS(R1);;;Reset free space counter
                                ;
                                ;  Clear out the output silo buffer
                                ;
002022  016500  000000G                 MOV     CL$ORB(R5),R0   ;;;Get pointer to start of output silo buffer
002026  010065  000000G                 MOV     R0,CL$ORP(R5)   ;;;Output character pointer
002032  010065  000000G                 MOV     R0,CL$ORG(R5)   ;;;Next available output character
002036  016565  000000G 000000G         MOV     CL$ORA(R5),CL$ORS(R5);;;Available space in output buffer
002044                                  ENABL                   ;** Enable interrupts **
                                ;
                                ;  Clear flag that says we have received an XOFF
                                ;
002052  042761  000000G 000000G         BIC     #$CTRLS,LSW3(R1);Say line output not suspended due to XOFF
                                ;
                                ;  Clear some status flags for the unit
                                ;
002060  042765  000000C 000000G         BIC     #<CM$WRT!CM$EFP!CM$CRL!CM$TBS!CM$EOF!CM$FFS>,CL$STA(R5)
                                ;
                                ;  If we are sending a break, stop now
                                ;
002066  032765  000000G 000000G         BIT     #CM$BRK,CL$STA(R5);Are we sending a break now?
002074  001406                          BEQ     1$              ;Br if not
002076  005000                          CLR     R0              ;Say to reset break transmission
002100  004737  006346'                 CALL    SETBRK          ;Stop sending break
002104  042765  000000G 000000G         BIC     #CM$BRK,CL$STA(R5);Say break transmission finished
                                ;
                                ;  Reset page and line position
                                ;
002112  005065  000000G         1$:     CLR     CL$LIN(R5)      ;Say we are at the top of a page
002116  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.
```

```
                                        ;
002122  032761  000000G 000000G         BIT     #$HISTP,LSW10(R1);Did we send an XOFF?
002130  001407                         BEQ     9$                ;Br if not
002132  042761  000000G 000000G         BIC     #$HISTP,LSW10(R1);Can XOFF has been cleared
002140  016100  000000G                 MOV     LCDTYP(R1),R0     ;Get line type index
002144  004770  000000G                 CALL    @CDSXON(R0)       ;Send XON
                                        ;
                                        ;  Finished
                                        ;
002150  012601                  9$:     MOV     (SP)+,R1
002152  000207                          RETURN
```

```
      1                                         .SBTTL  CLINCP -- Input character processing
      2                                 ;-----------------------------------------------------------------------
      3                                 ;  CLINCP is called at fork level after each received character has been
      4                                 ;  stored in the input silo buffer.  Its primary function is to move
      5                                 ;  characters from the input silo buffer to the user's data buffer.
      6                                 ;
      7                                 ;  Inputs:
      8                                 ;   R4 = Line index number of line that received a character.
      9                                 ;
     10 002154  010146                  CLINCP: MOV     R1,-(SP)
     11 002156  010546                          MOV     R5,-(SP)
     12                                 ;
     13                                 ;  Convert line index number to CL unit index
     14                                 ;
     15 002160  016405  000000G                 MOV     LCLUNT(R4),R5   ;Carry CL unit number in R5
     16                                 ;
     17                                 ;  If this CL unit is cross connected to a time-sharing line, try to
     18                                 ;  start output to the time-sharing line (it will fetch characters
     19                                 ;  directly from the input silo for the CL unit).
     20                                 ;
     21 002164  016501  000000G                 MOV     CL$XLN(R5),R1   ;Is this CL unit cross-connected to TT line?
     22 002170  001403                          BEQ     1$              ;Br if not
     23 002172  004777  000000G                 CALL    @TRNSTR         ;Try to start output to TT line
     24 002176  000402                          BR      9$
     25                                 ;
     26                                 ;  See if we need to move any characters from the input silo buffer
     27                                 ;  to the user's data buffer
     28                                 ;
     29 002200  004737  002212'         1$:     CALL    IRINGG          ;Move chars to user's data buffer
     30                                 ;
     31                                 ;  Finished
     32                                 ;
     33 002204  012605                  9$:     MOV     (SP)+,R5
     34 002206  012601                          MOV     (SP)+,R1
     35 002210  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
                                ;
002212                          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).
                                ;
002212                                  DISABL                  ;;;** Disable interrupts **
002220  032765  000000G 000000G         BIT     #CM$IRG,CL$STA(R5) ;;;Is this routine already active for unit?
002226  001404                          BEQ     2$              ;;;Br if not
002230                                  ENABL                   ;** Enable interrupts **
002236  000207                          RETURN
                                ;
                                ;  This routine is not active, claim it for us
                                ;
002240  052765  000000G 000000G 2$:     BIS     #CM$IRG,CL$STA(R5) ;;;Say the routine is now active
002246                                  ENABL                   ;** Enable interrupts **
                                ;
                                ;  Push some registers
                                ;
002254  010146                          MOV     R1,-(SP)
002256  010246                          MOV     R2,-(SP)
002260  010346                          MOV     R3,-(SP)
002262  010446                          MOV     R4,-(SP)
                                ;
                                ;  Get index number of line associated with this CL unit
                                ;
002264  016501  000000G                 MOV     CL$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.
                                ;
002270                          3$:     DISABL                  ;;;** Disable interrupts **
002276  016504  000000G                 MOV     CL$RQH(R5),R4   ;;;Is there a pending read request?
002302  001475                          BEQ     9$              ;;;Br if not
002304  032765  000000G 000000G         BIT     #CM$EOF,CL$STA(R5);;;Need to report end of file?
002312  001004                          BNE     7$              ;;;Br if yes
002314  026161  000000G 000000G         CMP     LHIRBS(R1),LHIRBA(R1);;;Any chars in the silo buffer?
002322  001465                          BEQ     9$              ;;;Br if not
                                ;
                                ;  There are characters in the silo buffer and there is a pending
                                ;  read request.
                                ;
002324                          7$:     ENABL                   ;** Enable interrupts **
                                ;
                                ;  See if flag is set which indicates that we should signal end-of-file
                                ;
002332  032765  000000G 000000G         BIT     #CM$EOF,CL$STA(R5);Should we signal end of file?
002340  001413                          BEQ     4$              ;Br if not
002342  016403  000000C                 MOV     Q.CSW-Q.BLKN(R4),R3;Get pointer to CSW for channel
002346  052763  000000G 000000G         BIS     #CS$EOF,C.CSW(R3);Set end of file flag
```

```
     58 002354  042765  000000G 000000G          BIC     #CM$EOF,CL$STA(R5);Acknowledge the EOF
     59 002362  004737  002742'                 CALL    RDFIN           ;Terminate this read operation
     60 002366  000740                          BR      3$              ;See if there is another read to do
     61                                 ;
     62                                 ;  Get a character from the silo buffer
     63                                 ;
     64 002370  004777  000000G         4$:     CALL    @SILFET         ;Get a character from input silo
     65 002374  103440                          BCS     9$              ;Br if no chars in silo
     66 002376  010002                          MOV     R0,R2           ;Get character to R2
     67                                 ;
     68                                 ;  If this is a control character, do special processing
     69                                 ;
     70 002400  020227  000032          6$:     CMP     R2,#32          ;Is this a control character?
     71 002404  101017                          BHI     5$              ;Br if not
     72 002406  105702                          TSTB    R2              ;Is this a null character?
     73 002410  001004                          BNE     8$              ;Br if not null
     74 002412  032765  000000G 000000G        BIT     #CO$BNI,CL$OPT(R5);Is binary input wanted?
     75 002420  001723                          BEQ     3$              ;Br if not -- ignore nulls
     76 002422  126427  000000C 000000G 8$:     CMPB    Q.FUNC-Q.BLKN(R4),#CLSFRB ;Is this a special read (.SPFUN 203)
     77 002430  001405                          BEQ     5$              ;If yes then accept control chars as normal
     78 002432  010200                          MOV     R2,R0           ;Get the control character
     79 002434  006300                          ASL     R0              ;Convert to word table index
     80 002436  004770  002524'                 CALL    @CCIRTN(R0)     ;Call control character processing routine
     81 002442  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 002444  004737  002706'         5$:     CALL    INPCHR          ;Store character into data buffer
     87                                 ;
     88                                 ;  If the input silo buffer is now empty, and this is a special function
     89                                 ;  read (.SPFUN 203), then say the read is finished.
     90                                 ;
     91 002450  126427  000000C 000000G        CMPB    Q.FUNC-Q.BLKN(R4),#CLSFRB ;is this a special read (.SPFUN 203)
     92 002456  001304                          BNE     3$              ;Br if not -- continue reading more
     93 002460  026161  000000G 000000G        CMP     LHIRBS(R1),LHIRBA(R1);Is the silo buffer empty?
     94 002466  001300                          BNE     3$              ;Br if not -- Get more chars for the SPFUN
     95 002470  004737  002742'                 CALL    RDFIN           ;Terminate the read operation
     96 002474  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 002476  042765  000000G 000000G 9$:     BIC     #CM$IRG,CL$STA(R5) ;;;Say we are leaving this routine
    102 002504                                  ENABL                   ;** Enable interrupts **
    103                                 ;
    104                                 ;  Finished
    105                                 ;
    106 002512  012604                          MOV     (SP)+,R4
    107 002514  012603                          MOV     (SP)+,R3
    108 002516  012602                          MOV     (SP)+,R2
    109 002520  012601                          MOV     (SP)+,R1
    110 002522  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
                                ;
002524  002620'         CCIRTN: .WORD   CCINUL           ;00 null
002526  002612'                 .WORD   CCISTR           ;01 SHO
002530  002612'                 .WORD   CCISTR           ;02 STX
002532  002612'                 .WORD   CCISTR           ;03 ETX
002534  002612'                 .WORD   CCISTR           ;04 EOT
002536  002612'                 .WORD   CCISTR           ;05 ENQ
002540  002612'                 .WORD   CCISTR           ;06 ACK
002542  002612'                 .WORD   CCISTR           ;07 BEL
002544  002612'                 .WORD   CCISTR           ;10 BACKSPACE
002546  002612'                 .WORD   CCISTR           ;11 TAB
002550  002632'                 .WORD   CCILF            ;12 LINE FEED
002552  002612'                 .WORD   CCISTR           ;13 VT
002554  002612'                 .WORD   CCISTR           ;14 FF
002556  002644'                 .WORD   CCICR            ;15 CARRIAGE RETURN
002560  002612'                 .WORD   CCISTR           ;16 SO
002562  002612'                 .WORD   CCISTR           ;17 SI
002564  002612'                 .WORD   CCISTR           ;20 DLE
002566  002612'                 .WORD   CCISTR           ;21 XON
002570  002612'                 .WORD   CCISTR           ;22 DC2
002572  002612'                 .WORD   CCISTR           ;23 XOFF
002574  002612'                 .WORD   CCISTR           ;24 DC4
002576  002612'                 .WORD   CCISTR           ;25 NAK
002600  002612'                 .WORD   CCISTR           ;26 SYN
002602  002612'                 .WORD   CCISTR           ;27 ETB
002604  002612'                 .WORD   CCISTR           ;30 CAN
002606  002612'                 .WORD   CCISTR           ;31 EM
002610  002672'                 .WORD   CCICTZ           ;32 SUB (ctrl-Z)
```

```
                                        ;
                                        ;  Routine to store the control character
                                        ;
002612  004737  002706'          CCISTR: CALL    INPCHR          ;Store the character
002616  000207                          RETURN
                                        ;
                                        ;  Routine to process a null character
                                        ;
002620  032765  000000G 000000G CCINUL: BIT     #CO$BNI,CL$OPT(R5);Are we in binary input mode?
002626  001371                          BNE     CCISTR          ;Br if yes -- go store the null
002630  000207                          RETURN                  ;Discard the null
                                        ;
                                        ;  Routine to process a line feed
                                        ;
002632  032765  000000G 000000G CCILF:  BIT     #CO$LFI,CL$OPT(R5);Should we ignore input line feeds?
002640  001364                          BNE     CCISTR          ;Br if not
002642  000207                          RETURN                  ;Discard the LF
                                        ;
                                        ;  Routine to process carriage returns
                                        ;
002644  016500  000000G          CCICR:  MOV     CL$RQH(R5),R0   ;Get address of current Q element
002650  126027  000000C 000000G         CMPB    Q.FUNC-Q.BLKN(R0),#CLSFRL ;Read-line special function?
002656  001355                          BNE     CCISTR          ;Br if not -- Treat CR as normal char
002660  004737  002706'                 CALL    INPCHR          ;Store the carriage return
002664  004737  002742'                 CALL    RDFIN           ;Terminate the read operation
002670  000207                          RETURN
                                        ;
                                        ;  Routine to process control-Z characters
                                        ;
002672                          CCICTZ:
                                        ;
                                        ;  Set flag which will cause us to return EOF status on next read
                                        ;
002672  052765  000000G 000000G         BIS     #CM$EOF,CL$STA(R5);Remember EOF has been hit
                                        ;
                                        ;  Terminate this read operation
                                        ;
002700  004737  002742'                 CALL    RDFIN           ;Terminate the read operation
002704  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
                                ;
002706  010446                  INPCHR: MOV     R4,-(SP)
                                ;
                                ;  Get address of current read queue element
                                ;
002710  016504  000000G         1$:     MOV     CL$RQH(R5),R4   ;Get pointer to current read queue element
002714  001410                          BEQ     9$              ;Br if no read request is pending
                                ;
                                ;  Store character into data buffer
                                ;
002716  010246                          MOV     R2,-(SP)        ;Stack the data char for PTBYT
002720  004737  000000G                 CALL    PTBYT           ;Move char to user's data buffer
                                ;
                                ;  Decrement remaining byte count and see if this completes the read request
                                ;
002724  005364  000000C                 DEC     Q.WCNT-Q.BLKN(R4);Does this complete the read request?
002730  001002                          BNE     9$              ;Br if not
                                ;
                                ;  The read request is completed.
                                ;  Return the queue element to the system.
                                ;
002732  004737  002742'                 CALL    RDFIN           ;Read request is completed
                                ;
                                ;  Finished
                                ;
002736  012604                  9$:     MOV     (SP)+,R4
002740  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.
                                ;
002742  010346                  RDFIN:  MOV     R3,-(SP)
002744  010446                          MOV     R4,-(SP)
                                ;
                                ;  Get address of current read queue element
                                ;
002746  016504  000000G                 MOV     CL$RQH(R5),R4   ;Get address of read queue element
002752  001427                          BEQ     9$              ;Br if none pending
                                ;
                                ;  See if we need to store nulls into the remainder of the buffer
                                ;
002754  016403  000000C                 MOV     Q.WCNT-Q.BLKN(R4),R3 ;Get remaining byte count
002760  001404                          BEQ     2$              ;Br if buffer is full
002762  005046                  1$:     CLR     -(SP)
002764  004737  000000G                 CALL    PTBYT           ;Null fill the remainder of the buffer
002770  077304                          SOB     R3,1$
                                ;
                                ;  Remove the queue element from our internal queue and place on the queue
                                ;  of elements waiting to be returned to the system.
                                ;
002772                          2$:     DISABL                  ;;;** Disable interrupts **
003000  016465  000000C 000000G         MOV     Q.LINK-Q.BLKN(R4),CL$RQH(R5) ;;;Remove Q element from list
003006  013764  000004' 000000C         MOV     CQH,Q.LINK-Q.BLKN(R4) ;;;Put Q element on completion list
003014  010437  000004'                 MOV     R4,CQH
003020                                  ENABL                   ;;; ** Enable interrupts **
                                ;
                                ;  Now call system I/O completion routine to free the queue element
                                ;
003026  004737  006010'                 CALL    RTNQ            ;Return queue element to the system
                                ;
                                ;  Finished
                                ;
003032  012604                  9$:     MOV     (SP)+,R4
003034  012603                          MOV     (SP)+,R3
003036  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.
                                ;
003040  010146                  CLTIMR: MOV     R1,-(SP)
003042  010446                          MOV     R4,-(SP)
003044  010546                          MOV     R5,-(SP)
                                ;
                                ;  Begin loop to service each CL unit
                                ;
003046  012705  000000C                 MOV     #2*<CLTOTL-1>,R5;Get index # of last CL unit
                                ;
                                ;  See if this CL unit is connected to a line
                                ;
003052  016501  000000G         1$:     MOV     CL$LIX(R5),R1   ;Is this CL unit connected to a line?
003056  001412                          BEQ     2$              ;Br if not
                                ;
                                ;  See if user wants to change status of Data Terminal Ready
                                ;
003060  004737  006250'                 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.
                                ;
003064  005765  000000G                 TST     CL$XLN(R5)      ;Is this CL unit cross connected to TT line?
003070  001403                          BEQ     3$              ;Br if not
003072  004737  004574'                 CALL    CLOCPY          ;Copy characters to CL output ring buffer
003076  000402                          BR      2$
003100  004737  003130'         3$:     CALL    ORINGP          ;Move chars to output ring buffer
                                ;
                                ;  Process the next CL unit
                                ;
003104                          2$:     ENABL                   ;Make sure interrupts are enabled
003112  162705  000002                  SUB     #2,R5           ;Get index of next line
003116  002355                          BGE     1$              ;Loop if more lines to service
                                ;
                                ;  Finished
                                ;
003120  012605                          MOV     (SP)+,R5
003122  012604                          MOV     (SP)+,R4
003124  012601                          MOV     (SP)+,R1
003126  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
                                ;
003130  010246                  ORINGP: MOV     R2,-(SP)
003132  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).
                                ;
003134                                  DISABL                  ;;;** Disable interrupts **
003142  032765  000000G 000000G         BIT     #CM$ORP,CL$STA(R5);;;Is this routine already active for unit?
003150  001402                          BEQ     21$             ;;;Br if not
003152  000137  003570'                 JMP     9$              ;;;Br if routine already active
                                ;
                                ;  This routine is not active for this unit.  Claim it.
                                ;
003156  052765  000000G 000000G 21$:    BIS     #CM$ORP,CL$STA(R5);;;Say routine is now active
003164                                  ENABL                   ;** Enable interrupts **
003172  005002                  11$:    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.
                                ;
003174                          4$:     DISABL                  ;** Disable interrupts **
003202  005765  000000G                 TST     CL$ORS(R5)      ;;;Any available space in ring buffer?
003206  001555                          BEQ     8$              ;;;Br if no space available
003210  005765  000000G                 TST     CL$WQH(R5)      ;;;Is there a pending write request?
003214  001004                          BNE     20$             ;;;Br if a write is pending
003216  032765  000000G 000000G         BIT     #CM$EFP,CL$STA(R5);;;Are we doing end-of-file processing?
003224  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.
                                ;
003226                          20$:    ENABL                   ;** Enable interrupts **
                                ;
                                ;  See if we are sending spaces to simulate tabs
                                ;
003234  032765  000000G 000000G 15$:    BIT     #CM$TBS,CL$STA(R5);Are we doing tab simulation?
003242  001412                          BEQ     16$             ;Br if not
003244  032765  000007  000000G         BIT     #7,CL$COL(R5)   ;Have we reached the next tab stop?
003252  001403                          BEQ     2$              ;Br if yes
003254  012700  000040                  MOV     #SPACE,R0       ;Get space for simulation
003260  000474                          BR      12$
003262  042765  000000G 000000G 2$:     BIC     #CM$TBS,CL$STA(R5);Say we are finished with tab simulation
                                ;
                                ;  See if we are sending line feeds to simulate a form feed
                                ;
003270  032765  000000G 000000G 16$:    BIT     #CM$FFS,CL$STA(R5);Are we doing form feed simulation?
003276  001414                          BEQ     1$              ;Br if not
```

```
     58 003300  026565  000000G 000000G          CMP     CL$LIN(R5),CL$LEN(R5) ;Have we reached top of new page yet?
     59 003306  103003                          BHIS    17$                   ;Br if yes
     60 003310  012700  000012                  MOV     #LF,R0                ;Send a line feed
     61 003314  000467                          BR      7$                    ;Go process the line feed
     62 003316  042765  000000G 000000G 17$:    BIC     #CM$FFS,CL$STA(R5);Say we have finished form feed simulation
     63 003324  005065  000000G                 CLR     CL$LIN(R5)            ;Say we are at top of new page
     64                                     ;
     65                                     ;  Try to get next character from user's data buffer
     66                                     ;
     67 003330  004737  003604' 000000G  1$:    CALL    GETCHR                ;Get next char from user's data buffer
     68 003334  103717                          BCS     4$                    ;Br if no chars left
     69                                     ;
     70                                     ;  Ignore user's FF immediately following FF from skip
     71                                     ;
     72 003336  032765  000000G 000000G          BIT     #CM$FFI,CL$STA(R5) ;Did we just do skip and should ignore FF?
     73 003344  001406                          BEQ     13$                   ;Br if not
     74 003346  042765  000000G 000000G          BIC     #CM$FFI,CL$STA(R5) ;Only ignore the 1st one
     75 003354  020027  000014                  CMP     R0,#FF                ;Is the 1st char after skip an FF?
     76 003360  001725                          BEQ     15$                   ;If yes, ignore this char
     77                                     ;
     78                                     ;  See if this is a control character
     79                                     ;
     80 003362  032765  000000G 000000G 13$:    BIT     #CO$BNO,CL$OPT(R5);Are we in binary output mode?
     81 003370  001046                          BNE     5$                    ;Br if yes -- Accept all chars
     82 003372  032765  000000G 000000G          BIT     #CO$8BT,CL$OPT(R5);Is 8 bit support wanted?
     83 003400  001002                          BNE     18$                   ;Br if yes
     84 003402  042700  177600                  BIC     #^C<177>,R0           ;Mask character to 7 bits
     85 003406  020027  000037          18$:    CMP     R0,#37                ;Is this a control character?
     86 003412  101430                          BLOS    7$                    ;Br if yes
     87 003414  042765  000000G 000000G          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 003422  032765  000000G 000000G          BIT     #CO$LC,CL$OPT(R5);May we send lower-case characters?
     93 003430  001010                          BNE     12$                   ;Br if yes
     94 003432  020027  000141                  CMP     R0,#141               ;Is this a lower-case letter?
     95 003436  103405                          BLO     12$                   ;Br if not
     96 003440  120027  000172                  CMPB    R0,#172
     97 003444  101002                          BHI     12$
     98 003446  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 003452  005265  000000G          12$:    INC     CL$COL(R5)            ;Advance column counter
    103 003456  016503  000000G                  MOV     CL$WID(R5),R3         ;Was a WIDTH parameter specified?
    104 003462  001411                          BEQ     5$                    ;Br if not
    105 003464  026503  000000G                  CMP     CL$COL(R5),R3         ;Have we reached the specified width?
    106 003470  101406                          BLOS    5$                    ;Br if not
    107 003472  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 003474  010003                  7$:     MOV     R0,R3                 ;Get control character
    113 003476  006303                          ASL     R3                    ;Convert to word table index
    114 003500  004773  004052'                 CALL    @CCORTN(R3)           ;Call processing routine
```

```
    115 003504  103653                          BCS     15$             ;Br if we should discard this character
    116                                 ;
    117                                 ;  Move character to output ring buffer
    118                                 ;
    119 003506  016503  000000G         5$:     MOV     CL$ORP(R5),R3   ;Get position for char in ring buffer
    120 003512  110023                          MOVB    R0,(R3)+        ;Store char into ring buffer
    121                                 ;
    122                                 ;  Say 1 less free char space in ring buffer
    123                                 ;
    124 003514  005365  000000G                 DEC     CL$ORS(R5)      ;One less free char pos in out ring buffer
    125 003520  005202                          INC     R2              ;Count # chars moved to ring buffer
    126                                 ;
    127                                 ;  Save updated ring buffer pointer
    128                                 ;
    129 003522  020365  000000G                 CMP     R3,CL$ORE(R5)   ;Did we advance past end of ring buffer?
    130 003526  103402                          BLO     6$              ;Br if not
    131 003530  016503  000000G                 MOV     CL$ORB(R5),R3   ;Wrap around to front of ring buffer
    132 003534  010365  000000G         6$:     MOV     R3,CL$ORP(R5)   ;Save new ring buffer pointer
    133 003540  000615                          BR      4$              ;Go see if we should send more chars
    134                                 ;
    135                                 ;  Finished moving characters to output ring buffer.
    136                                 ;  If we moved any characters, call the routine to try to start output
    137                                 ;  to the line.
    138                                 ;
    139 003542  005702                  8$:     TST     R2              ;;;Did we move any characters to ring buffer?
    140 003544  001406                          BEQ     10$             ;;;Br if not
    141 003546                                  ENABL                   ;** Enable interrupts **
    142 003554  004737  005514'                 CALL    CLSTRT          ;Try to start transmission to this line
    143 003560  000604                          BR      11$             ;Go back and check for more to send
    144                                 ;
    145                                 ;  Release this routine for this unit
    146                                 ;
    147 003562  042765  000000G 000000G 10$:    BIC     #CM$ORP,CL$STA(R5);;;Say routine is now free
    148                                 ;
    149                                 ;  Finished
    150                                 ;
    151 003570                          9$:     ENABL                   ;** Enable interrupts **
    152 003576  012603                          MOV     (SP)+,R3
    153 003600  012602                          MOV     (SP)+,R2
    154 003602  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
                                ;
003604  010446                  GETCHR: MOV     R4,-(SP)
                                ;
                                ;  See if we should do end-of-file output processing.
                                ;
003606  032765  000000G 000000G 5$:     BIT     #CM$EFP,CL$STA(R5) ;Should we do end-of-file processing?
003614  001403                          BEQ     2$                 ;Br if not
                                ;
                                ;  We are doing end-of-file output processing.
                                ;  See if there is another end-of-file character to send.
                                ;
003616  004737  003760'                 CALL    EOFCHR             ;See if another eof char to send
003622  103054                          BCC     12$                ;Br if we got an EOF character
                                ;
                                ;  See if there is a pending write operation
                                ;
003624  016504  000000G         2$:     MOV     CL$WQH(R5),R4      ;Get pointer to current write queue element
003630  001446                          BEQ     10$                ;Br if no pending write operation
                                ;
                                ;  If the FORM0 option is in effect and this is the first write to
                                ;  block 0, send a form feed.
                                ;
003632  005764  000000C                 TST     Q.BLKN-Q.BLKN(R4);Is block number = 0?
003636  001011                          BNE     4$                 ;Br if not
003640  032765  000000G 000000G        BIT     #CO$FF0,CL$OPT(R5) ;Is the FORM0 option in effect?
003646  001405                          BEQ     4$                 ;Br if not
003650  005264  000000C                 INC     Q.BLKN-Q.BLKN(R4);Inc block # so we only do this once
003654  112700  000014                 MOVB    #FF,R0             ;Get form feed character
003660  000434                          BR      9$                 ;Return the form feed
                                ;
                                ;  See if current queue element has another character to be sent
                                ;
003662  005764  000000C         4$:     TST     Q.WCNT-Q.BLKN(R4);Any remaining bytes to send?
003666  001406                          BEQ     3$                 ;Br if not -- write request is finished
                                ;
                                ;  Get next character from user's buffer
                                ;
003670  005364  000000C                 DEC     Q.WCNT-Q.BLKN(R4);Decrease remaining byte count
003674  004737  000000G                 CALL    GTBYT              ;Get next byte from user's buffer
003700  012600                          MOV     (SP)+,R0           ;Get the returned character
003702  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.
```

```
                                        ;
003704                          3$:     DISABL                  ;;;** Disable interrupts **
003712  016465  000000C 000000G        MOV     Q.LINK-Q.BLKN(R4),CL$WQH(R5) ;;;Remove element from internal Q
003720  013764  000004' 000000C        MOV     CQH,Q.LINK-Q.BLKN(R4) ;;;Add to list of completed requests
003726  010437  000004'                MOV     R4,CQH
003732                                  ENABL                   ;** Enable interrupts **
                                        ;
                                        ;  Return the completed queue element to the system (do .DRFIN)
                                        ;
003740  004737  006010'                 CALL    RTNQ            ;Tell system we finished the operation
                                        ;
                                        ;  Go back and see if there is another write request pending
                                        ;
003744  000727                          BR      2$              ;Go check for another write request
                                        ;
                                        ;  There are no available characters
                                        ;
003746  000261                  10$:    SEC                     ;Signal that no chars are available
003750  000401                          BR      12$
                                        ;
                                        ;  We got a character
                                        ;
003752  000241                  9$:     CLC                     ;Signal that we got a character
                                        ;
                                        ;  Finished
                                        ;
003754  012604                  12$:    MOV     (SP)+,R4
003756  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.
                                ;
003760                          EOFCHR:
                                ;
                                ;  See if we need to send form-feeds
                                ;
003760  126565  000001G 000000G         CMPB    CL$EPN+1(R5),CL$EPN(R5);Do we need to send more form-feeds?
003766  103005                         BHIS    1$                      ;Br if not
003770  105265  000001G                INCB    CL$EPN+1(R5)            ;Count another form-feed being sent
003774  012700  000014                 MOV     #FF,R0                  ;Get form-feed character
004000  000422                         BR      7$                      ;Go send it
                                ;
                                ;  See if we need to send characters from ENDSTRING
                                ;
004002  016500  000000G         1$:     MOV     CL$EPP(R5),R0           ;Are we sending end-string characters?
004006  001405                         BEQ     2$                      ;Br if not
004010  111000                         MOVB    (R0),R0                 ;Get next char to send
004012  001403                         BEQ     2$                      ;Br if reached end of string
004014  005265  000000G                INC     CL$EPP(R5)              ;Advance character pointer
004020  000412                         BR      7$                      ;Go send the character
                                ;
                                ;  We have finished all end-of-file output processing
                                ;
004022  105065  000001G         2$:     CLRB    CL$EPN+1(R5)            ;Reset form-feed count
004026  016565  000000G 000000G        MOV     CL$EPS(R5),CL$EPP(R5);Reset end-string pointer
004034  042765  000000G 000000G        BIC     #CM$EFP,CL$STA(R5);Finished end-of-file output processing
004042  000261                         SEC                             ;Signal that no character was gotten
004044  000401                         BR      9$
                                ;
                                ;  We got a character
                                ;
004046  000241                 7$:     CLC                             ;Signal that we got a character
                                ;
                                ;  Finished
                                ;
004050  000207                 9$:     RETURN
```

```
      1                                       .SBTTL  CCORTN -- Output control character processing routines
      2                                 ;-----------------------------------------------------------------------
      3                                 ;  Processing routines for output control characters.
      4                                 ;  When one of these routines is called, R0 contains the control character.
      5                                 ;  If the character is to be sent, the C-flag is cleared on return.
      6                                 ;  If the character is to be discarded, the C-flag is set on return.
      7                                 ;
      8                                 ;  Vector of control character processing routines
      9                                 ;
     10 004052  004200'                 CCORTN: .WORD   CCONUL           ;00 null
     11 004054  004152'                         .WORD   CCOCTL           ;01 SHO
     12 004056  004152'                         .WORD   CCOCTL           ;02 STX
     13 004060  004152'                         .WORD   CCOCTL           ;03 ETX
     14 004062  004152'                         .WORD   CCOCTL           ;04 EOT
     15 004064  004152'                         .WORD   CCOCTL           ;05 ENQ
     16 004066  004152'                         .WORD   CCOCTL           ;06 ACK
     17 004070  004152'                         .WORD   CCOCTL           ;07 BEL
     18 004072  004204'                         .WORD   CCOBS            ;10 BACKSPACE
     19 004074  004226'                         .WORD   CCOTAB           ;11 TAB
     20 004076  004320'                         .WORD   CCOLF            ;12 LINE FEED
     21 004100  004152'                         .WORD   CCOCTL           ;13 VT
     22 004102  004454'                         .WORD   CCOFF            ;14 FF
     23 004104  004530'                         .WORD   CCOCR            ;15 CARRIAGE RETURN
     24 004106  004152'                         .WORD   CCOCTL           ;16 SO
     25 004110  004152'                         .WORD   CCOCTL           ;17 SI
     26 004112  004152'                         .WORD   CCOCTL           ;20 DLE
     27 004114  004152'                         .WORD   CCOCTL           ;21 DC1 (ctrl-Q)
     28 004116  004152'                         .WORD   CCOCTL           ;22 DC2
     29 004120  004152'                         .WORD   CCOCTL           ;23 DC3 (ctrl-S)
     30 004122  004152'                         .WORD   CCOCTL           ;24 DC4
     31 004124  004152'                         .WORD   CCOCTL           ;25 NAK
     32 004126  004152'                         .WORD   CCOCTL           ;26 SYN
     33 004130  004152'                         .WORD   CCOCTL           ;27 ETB
     34 004132  004152'                         .WORD   CCOCTL           ;30 CAN
     35 004134  004152'                         .WORD   CCOCTL           ;31 EM
     36 004136  004152'                         .WORD   CCOCTL           ;32 SUB (ctrl-Z)
     37 004140  004152'                         .WORD   CCOCTL           ;33 ESC
     38 004142  004152'                         .WORD   CCOCTL           ;34 FS
     39 004144  004152'                         .WORD   CCOCTL           ;35 GS
     40 004146  004152'                         .WORD   CCOCTL           ;36 RS
     41 004150  004152'                         .WORD   CCOCTL           ;37 US
```

```
                                        ;
                                        ;  Process a general control character
                                        ;
004152  042765  000000G 000000G CCOCTL: BIC     #CM$CRL,CL$STA(R5) ;Say last char out was not carriage return
004160  032765  000000G 000000G        BIT     #CO$CTL,CL$OPT(R5) ;Are we to transmit control chars?
004166  001002                       BNE     CCOSND          ;Br if yes
004170  000261                       SEC                     ;Say to ignore this character
004172  000207                       RETURN
                                        ;
                                        ;  Routine to cause the current control character to be transmitted unchanged
                                        ;
004174  000241                   CCOSND: CLC                     ;Say to send the character
004176  000207                       RETURN
                                        ;
                                        ;  Process null character
                                        ;
004200  000261                   CCONUL: SEC                     ;Say to ignore this character
004202  000207                       RETURN
                                        ;
                                        ;  Proccess Backspace character
                                        ;
004204  042765  000000G 000000G CCOBS:  BIC     #CM$CRL,CL$STA(R5) ;Say last char out was not carriage return
004212  005365  000000G                DEC     CL$COL(R5)      ;Say we are moving back 1 char
004216  002366                       BGE     CCOSND          ;Br if did not go past column 0
004220  005065  000000G                CLR     CL$COL(R5)      ;Constrain to column 0
004224  000763                       BR      CCOSND          ;Go send the character
                                        ;
                                        ;  Process tab character
                                        ;
004226  042765  000000G 000000G CCOTAB: BIC     #CM$CRL,CL$STA(R5) ;Say last char out was not carriage return
004234  032765  000000G 000000G        BIT     #CO$TAB,CL$OPT(R5) ;Does device have hardware tab support
004242  001416                       BEQ     1$              ;Br if not
004244  062765  000010  000000G        ADD     #8.,CL$COL(R5)  ;Bound up to next tab stop
004252  042765  000007  000000G        BIC     #7,CL$COL(R5)
004260  005765  000000G                TST     CL$WID(R5)      ;Was a maximum width specified?
004264  001743                       BEQ     CCOSND          ;Br if not -- go send the tab
004266  026565  000000G 000000G        CMP     CL$COL(R5),CL$WID(R5) ;Have we gone beyond max width?
004274  103737                       BLO     CCOSND          ;Br if not
004276  000740                       BR      CCONUL          ;Discard this tab
004300  052765  000000G 000000G 1$:    BIS     #CM$TBS,CL$STA(R5) ;Say we are doing tab simulation
004306  005265  000000G                INC     CL$COL(R5)      ;Advance column counter
004312  012700  000040                MOV     #SPACE,R0       ;Send a space character
004316  000726                       BR      CCOSND
                                        ;
                                        ;  Process Line feed character
                                        ;
004320  005265  000000G          CCOLF: INC     CL$LIN(R5)      ;Increment line-on-page counter
004324  016500  000000G                MOV     CL$LEN(R5),R0   ;Was a page length value specified?
004330  001431                       BEQ     5$              ;Br if not
004332  026500  000000G                CMP     CL$LIN(R5),R0   ;Have we reached the top of a new page?
004336  103405                       BLO     2$              ;Br if not
004340  005065  000000G                CLR     CL$LIN(R5)      ;Say we are at top of a new page
004344  042765  000000G 000000G        BIC     #CM$FFS,CL$STA(R5);Stop doing form feed simulation
004352  166500  000000G          2$:   SUB     CL$SKP(R5),R0   ;See if we are to skip lines at bottom of page
004356  026500  000000G                CMP     CL$LIN(R5),R0   ;Have we reached the skip point?
004362  001014                       BNE     5$              ;Br if not
004364  032765  000000G 000000G        BIT     #CM$FFS,CL$STA(R5);Are we already doing form feed simulation?
```

```
                                        BNE     5$              ;Br if yes
                                        MOVB    #FF,R0          ;At skip point -- Do a form feed
                                        BIS     #CM$FFI,CL$STA(R5) ;Ignore FF if 1st char after skip
                                        DEC     CL$LIN(R5)      ;Set line counter back -- haven't sent LF yet
                                        BR      CCOFF           ;Go process the form feed
                                5$:     BIT     #CO$LFO,CL$OPT(R5) ;Should we discard line feeds on output?
                                        BNE     6$              ;Br if not
                                        BIT     #CM$CRL,CL$STA(R5);Was last char out a carriage return?
                                        BEQ     6$              ;Br if not
                                        BIC     #CM$CRL,CL$STA(R5);Clear flag that says carriage return last
                                        BR      CCONUL          ;Discard the line feed
                                6$:     MOVB    #LF,R0          ;Get back line feed character
                                        CLC                     ;Say to send it
                                9$:     RETURN
                                ;
                                ;  Process Form feed character
                                ;
                                CCOFF:  BIC     #CM$CRL,CL$STA(R5) ;Say last char out was not carriage return
                                        BIT     #CO$FF,CL$OPT(R5) ;Does this device support form feed chars?
                                        BEQ     1$              ;Br if not
                                        CLR     CL$LIN(R5)      ;Say we are at top of the page
                                        BR      CCOSND          ;Go send the form feed
                                1$:     TST     CL$LEN(R5)      ;Do we have a non-zero page length?
                                        BEQ     2$              ;If not then discard the FF
                                        BIS     #CM$FFS,CL$STA(R5);Say we are starting form-feed simulation
                                        MOV     #LF,R0          ;Translate form feed to line feed
                                        BR      CCOLF           ;Go send line feed
                                2$:     CLR     CL$LIN(R5)      ;Say we are at top of page
                                        BR      CCONUL          ;Discard the character
                                ;
                                ;  Process carriage return character
                                ;
                                CCOCR:  BIS     #CM$CRL,CL$STA(R5) ;Say last char out was carriage return
                                        CLR     CL$COL(R5)      ;Say we are back to column 0
                                        BIT     #CO$CR,CL$OPT(R5);Should we transmit carriage returns?
                                        BNE     CCOSND          ;Br if yes
                                        SEC                     ;Ignore this char
                                        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
                                ;
004556  010546                  CLXICP: MOV     R5,-(SP)
                                ;
                                ;  Get CL index of line we are cross connected to
                                ;
004560  016105  000000G                 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
                                ;
004564  004737  004574'                 CALL    CLOCPY          ;Copy chars to CL output silo
                                ;
                                ;  Finished
                                ;
004570  012605                          MOV     (SP)+,R5
004572  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
                                ;
004574  010146                  CLOCPY: MOV     R1,-(SP)
004576  010246                          MOV     R2,-(SP)
004600  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).
                                ;
004602                                  DISABL                  ;;;Disable interrupts
004610  032765  000000G 000000G         BIT     #CM$ORP,CL$STA(R5);;;Is this routine already active?
004616  001113                          BNE     9$              ;;;Br if yes
                                ;
                                ;  This routine is not active for this unit.  Claim it.
                                ;
004620  052765  000000G 000000G         BIS     #CM$ORP,CL$STA(R5);;;Say routine is now active
004626                                  ENABL                   ;Enable interrupts
004634  005002                  11$:    CLR     R2              ;Count # chars copied to output buffer
                                ;
                                ;  See if cross-connection is still in effect
                                ;
004636  016501  000000G         4$:     MOV     CL$XLN(R5),R1   ;Get number of cross-connected TT line
004642  001476                          BEQ     10$             ;Br if no longer cross connected
                                ;
                                ;  See if there is any free space in the output ring buffer.
                                ;
004644                                  DISABL                  ;;;Disable interrupts
004652  005765  000000G                 TST     CL$ORS(R5)      ;;;Any available space in ring buffer?
004656  001460                          BEQ     8$              ;;;Br if no space available
004660  026161  000000G 000000G         CMP     LHIRBS(R1),LHIRBA(R1);;;Any chars in TT input silo?
004666  001454                          BEQ     8$              ;;;Br if not
004670                                  ENABL                   ;Enable interrupts
                                ;
                                ;  Get next character from TT input silo
                                ;
004676  004777  000000G                 CALL    @SILFET         ;Get next char from TT input silo
004702  103446                          BCS     8$              ;Br if no more chars available
                                ;
                                ;  We got a character.
                                ;  See if character has special significance.
                                ;
004704  032765  000000G 000000G         BIT     #CM$MCC,CL$STA(R5);Modem control or literal char?
004712  001407                          BEQ     1$              ;Br if not
004714  042765  000000G 000000G         BIC     #CM$MCC,CL$STA(R5);Reset literal-character flag
004722  004737  005064'                 CALL    CLXMCC          ;Process the character
004726  103743                          BCS     4$              ;Br if finished with char
004730  000415                          BR      2$              ;Go transmit the character
004732  120037  000000G         1$:     CMPB    R0,VCXTRM       ;Control-\ -- Terminate connection?
004736  001003                          BNE     3$              ;Br if not
004740  004737  005412'                 CALL    CLXBRK          ;Break cross connection and drop DTR
```

```
     58 004744  000425                          BR      8$                ;Finished
     59 004746  120037  000000G          3$:     CMPB    R0,VCXCTL         ;Control-A means next char is modem control
     60 004752  001004                          BNE     2$                ;Br if not ctrl-A
     61 004754  052765  000000G 000000G         BIS     #CM$MCC,CL$STA(R5);Remember next char is modem control
     62 004762  000725                          BR      4$                ;Go get next char
     63                                    ;
     64                                    ;  Store this character into the output ring buffer
     65                                    ;
     66 004764  016503  000000G          2$:     MOV     CL$ORP(R5),R3     ;Get position for char in ring buffer
     67 004770  110023                          MOVB    R0,(R3)+          ;Store char into ring buffer
     68                                    ;
     69                                    ;  Count chars in ring buffer
     70                                    ;
     71 004772  005202                          INC     R2                ;One more char stored into ring buffer
     72 004774  005365  000000G                 DEC     CL$ORS(R5)        ;One less free space in ring buffer
     73                                    ;
     74                                    ;  Save updated ring buffer pointer
     75                                    ;
     76 005000  020365  000000G                 CMP     R3,CL$ORE(R5)     ;Did we advance past end of ring buffer?
     77 005004  103402                          BLO     6$                ;Br if not
     78 005006  016503  000000G                 MOV     CL$ORB(R5),R3     ;Wrap around to front of ring buffer
     79 005012  010365  000000G          6$:     MOV     R3,CL$ORP(R5)     ;Save new ring buffer pointer
     80 005016  000707                          BR      4$                ;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 005020  005702                   8$:     TST     R2                ;;;Did we copy any characters?
     88 005022  001406                          BEQ     10$               ;;;Br if not
     89 005024                                  ENABL                     ;Enable interrupts
     90 005032  004737  005514'                 CALL    CLSTRT            ;Start transmission to CL line
     91 005036  000676                          BR      11$               ;Go back and try to copy more
     92                                    ;
     93                                    ;  Release this routine for this unit
     94                                    ;
     95 005040  042765  000000G 000000G  10$:    BIC     #CM$ORP,CL$STA(R5);Say routine is now free
     96                                    ;
     97                                    ;  Finished
     98                                    ;
     99 005046                           9$:     ENABL                     ;Enable interrupts
    100 005054  012603                          MOV     (SP)+,R3
    101 005056  012602                          MOV     (SP)+,R2
    102 005060  012601                          MOV     (SP)+,R1
    103 005062  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.
                                ;
005064  010046                  CLXMCC: MOV     R0,-(SP)        ;Save the character
005066  010146                          MOV     R1,-(SP)
                                ;
                                ;  Translate lower-case to upper-case
                                ;
005070  120027  000141                  CMPB    R0,#141         ;Is this a lower-case letter?
005074  103405                          BLO     1$              ;Br if not
005076  120027  000172                  CMPB    R0,#172
005102  101002                          BHI     1$              ;Br if not
005104  162700  000040                  SUB     #40,R0          ;Convert to upper-case
                                ;
                                ;  "B" -- Start sending a break
                                ;
005110  120027  000102          1$:     CMPB    R0,#'B          ;Is character B?
005114  001042                          BNE     2$              ;Br if not
005116  052765  000000G 000000G         BIS     #CM$BRK,CL$STA(R5);Set flag saying we are sending break
005124  012700  000000G                 MOV     #MS$BRK,R0      ;Set flag to start break transmission
005130  004737  006346'                 CALL    SETBRK          ;Call hardware routine to start sending break
005134  004737  005514'                 CALL    CLSTRT          ;Start transmitter
005140  004737  000000G                 CALL    GETRTQ          ;Get a real-time queue element (ptr in R1)
005144  012761  000036  000000G         MOV     #30.,CQ$LOT(R1) ;Set approx 0.5 second time interval
005152  012761  005362' 000000G         MOV     #CLXSSB,CQ$RTN(R1);Set address of compl routine
005160  013761  000000G 000000G         MOV     @#KPAR5,CQ$PA5(R1);Save system par 5 mapping
005166  010561  000000G                 MOV     R5,CQ$RO(R1)    ;Set CL unit index
005172                                  DISABL                  ;;;* Disable interrupts *
005200  013761  000000G 000000G         MOV     MRKTHD,CQ$LNK(R1);;;Put new element on linked list
005206  010137  000000G                 MOV     R1,MRKTHD       ;;;
005212                                  ENABL                   ;* Enable interrupts *
005220  000452                          BR      20$
                                ;
                                ;  "D" -- Raise DTR
                                ;
005222  120027  000104          2$:     CMPB    R0,#'D          ;Is character D?
005226  001006                          BNE     4$              ;Br if not
005230  052765  000000G 000000G         BIS     #CO$DTR,CL$OPT(R5);Request DTR up
005236  004737  006250'                 CALL    SETDTR          ;Call routine to raise DTR
005242  000441                          BR      20$
                                ;
                                ;  "H" -- Drop DTR
                                ;
005244  120027  000110          4$:     CMPB    R0,#'H          ;Is character H?
005250  001006                          BNE     5$              ;Br if not
005252  042765  000000G 000000G         BIC     #CO$DTR,CL$OPT(R5);Request DTR drop
005260  004737  006250'                 CALL    SETDTR          ;Call routine to drop DTR
005264  000430                          BR      20$
```

```
                                        ;
                                        ;  "R" -- Reset XON/XOFF status
                                        ;
005266  120027  000122         5$:     CMPB    R0,#'R          ;Reset XON/XOFF status?
005272  001017                         BNE     6$              ;Br if not
005274  016501  000000G                MOV     CL$LIX(R5),R1   ;Get index of line we are connected to
005300  042761  000000G 000000G        BIC     #$CTRLS,LSW3(R1);Reset XOFF received flag
005306  042761  000000G 000000G        BIC     #$HISTP,LSW10(R1);Say input has not been stopped by XOFF
005314  016100  000000G                MOV     LCDTYP(R1),R0   ;Get device type code
005320  004770  000000G                CALL    @CDSXON(R0)     ;Call routine to stuff XON into output
005324  004737  005514'                CALL    CLSTRT          ;Try to start output to CL unit
005330  000406                         BR      20$
                                        ;
                                        ;  "X" -- Break cross connection without dropping DTR
                                        ;
005332  120027  000130         6$:     CMPB    R0,#'X          ;Break cross-connection?
005336  001005                         BNE     21$             ;Br if not
005340  004737  005432'                CALL    CLXDRP          ;Break cross connection
005344  000400                         BR      20$             ;Finished with character
                                        ;
                                        ;  This is a modem control character.
                                        ;  Don't send it.
                                        ;
005346  000261                 20$:    SEC                     ;Signal not to send the character
005350  000401                         BR      22$
                                        ;
                                        ;  This is not a modem control character.
                                        ;  Send the character literally.
                                        ;
005352  000241                 21$:    CLC                     ;Signal that we should send the character
                                        ;
                                        ;  Finished
                                        ;
005354  012601                 22$:    MOV     (SP)+,R1
005356  012600                         MOV     (SP)+,R0
005360  000207                         RETURN
```

```
                                ;-----------------------------------------------------------------------
                                ;  System completion routine called to stop sending break to a
                                ;  cross-connected CL line.
                                ;
                                ;  Inputs:
                                ;   R0 = CL unit index
                                ;
005362  010546                  CLXSSB: MOV     R5,-(SP)
005364  010005                          MOV     R0,R5           ;Get CL unit index
                                ;
                                ;  Stop sending break
                                ;
005366  005000                          CLR     R0              ;Clear break-send flag
005370  004737  006346'                 CALL    SETBRK          ;Call hardware routine to end break
005374  042765  000000G 000000G         BIC     #CM$BRK,CL$STA(R5);Clear break-sending flag
005402  004737  005514'                 CALL    CLSTRT          ;Start transmitter
                                ;
                                ;  Finished
                                ;
005406  012605                          MOV     (SP)+,R5
005410  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
                                ;
005412                          CLXBRK:
                                ;
                                ;  First, drop DTR
                                ;
005412  042765  0000000 0000000         BIC     #CO$DTR,CL$OPT(R5) ;Request DTR drop
005420  004737  006250'                 CALL    SETDTR          ;Call routine to drop DTR
                                ;
                                ;  Now break the cross connection
                                ;
005424  004737  005432'                 CALL    CLXDRP          ;Break the cross connection
                                ;
                                ;  Finished
                                ;
005430  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
                                ;
005432  010146                  CLXDRP: MOV     R1,-(SP)
                                ;
                                ;  Reset this CL unit
                                ;
005434  004737  001764'                 CALL    CLREST          ;Reset the CL unit
                                ;
                                ;  Reconnect time-sharing line to normal input character processing routine
                                ;
005440                                  DISABL                  ;;;** Disable interrupts **
005446  016501  000000G                 MOV     CL$XLN(R5),R1   ;;;Get number of cross-connected TT line
005452  012761  177777  000000G         MOV     #-1,LXCL(R1)    ;;;Say not connected to a CL unit
005460  012761  000000G 000000G         MOV     #TTINCP,LINIR(R1);;;Connect to TT input processing routine
                                ;
                                ;  Say CL unit no longer connected to time-sharing line
                                ;
005466  005065  000000G                 CLR     CL$XLN(R5)      ;;;CL unit no longer connected to TT line
005472                                  ENABL                   ;** Enable interrupts **
                                ;
                                ;  Restart the execution of the job
                                ;
005500  116101  000000G                 MOVB    LNMAP(R1),R1    ;Get virtual job index number
005504  004737  000000G                 CALL    FORCEX          ;Cause job to continue execution
                                ;
                                ;  Finished
                                ;
005510  012601                          MOV     (SP)+,R1
005512  000207                          RETURN
```

```
                                        .SBTTL  CLSTRT -- Start transmissions to a line
                                ;-----------------------------------------------------------------------
                                ;  CLSTRT is called to initiate transmission to a line.
                                ;
                                ;  Inputs:
                                ;   R5 = CL unit index number.
                                ;
005514  010146                  CLSTRT: MOV     R1,-(SP)
                                ;
                                ;  Convert CL unit number into line index number
                                ;
005516  016501  000000G                 MOV     CL$LIX(R5),R1   ;Get line index # for this CL unit
                                ;
                                ;  Call device dependent routine to start the transmitter
                                ;
005522  016100  000000G                 MOV     LCDTYP(R1),R0   ;Get communications device type code
005526  004770  000000G                 CALL    @CDSTRT(R0)     ;Call device dependent startup routine
005532  005237  000000G                 INC     NEDCDO          ;Say output character processing needed
005536  005237  000000G                 INC     NEDCLO          ;Say CL output processing needed
                                ;
                                ;  Finished
                                ;
005542  012601                          MOV     (SP)+,R1
005544  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
                                ;
005546  010346                  CLABRT: MOV     R3,-(SP)
005550  010446                          MOV     R4,-(SP)
005552  010546                          MOV     R5,-(SP)
                                ;
                                ;  Check each CL unit to see if there are any requests for this job
                                ;
005554  012705  000000C                 MOV     #2*<CLTOTL-1>,R5;Get index to last CL unit
005560  012703  000000G          1$:    MOV     #CL$RQH,R3      ;Get address of read queue head
005564  004737  005622'                  CALL    CKABTQ          ;See if there are any entries on this queue
005570  012703  000000G                 MOV     #CL$WQH,R3      ;Get address of write queue head
005574  004737  005622'                  CALL    CKABTQ          ;See if there are any entries on this queue
005600  162705  000002                  SUB     #2,R5           ;Get index number of next CL unit
005604  002365                          BGE     1$              ;Br if more units to check
                                ;
                                ;  Call routine to return any freed queue elements to the system
                                ;
005606  004737  006010'                  CALL    RTNQ            ;Return freed queue elements to the system
                                ;
                                ;  Finished
                                ;
005612  012605                          MOV     (SP)+,R5
005614  012604                          MOV     (SP)+,R4
005616  012603                          MOV     (SP)+,R3
005620  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.
                                ;
005622  010246                  CKABTQ: MOV     R2,-(SP)
005624  010346                          MOV     R3,-(SP)
005626  010546                          MOV     R5,-(SP)
                                ;
                                ;  Get address of queue head
                                ;
005630  060305                          ADD     R3,R5           ;Point to queue head for this unit
005632  162705  000000C               SUB     #Q.LINK-Q.BLKN,R5;Make head look like fake queue entry
                                ;
                                ;  Search for entries in the queue
                                ;
005636  010503                  1$:     MOV     R5,R3           ;Point to queue head
005640                                  DISABL                  ;;;** Disable interrupts **
005646  010302                          MOV     R3,R2           ;;;Save address of current entry
005650  016303  000000C         2$:     MOV     Q.LINK-Q.BLKN(R3),R3;;;Get address of next entry
005654  001417                          BEQ     9$              ;;;Br if no entries for job being aborted
005656  120463  000000C               CMPB    R4,Q.JOB-Q.BLKN(R3);;;Is this the job being aborted?
005662  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
                                ;
005664  016362  000000C 000000C       MOV     Q.LINK-Q.BLKN(R3),Q.LINK-Q.BLKN(R2);;;Remove from list
005672  013763  000004' 000000C       MOV     CQH,Q.LINK-Q.BLKN(R3);;;Put on completion list
005700  010337  000004'               MOV     R3,CQH
                                ;
                                ;  Go back and see if there are any more entries to remove
                                ;
005704                                  ENABL                   ;** Enable interrupts **
005712  000751                          BR      1$              ;Go repeat the process
                                ;
                                ;  Finished with this queue
                                ;
005714                          9$:     ENABL                   ;** Enable interrupts **
005722  012605                          MOV     (SP)+,R5
005724  012603                          MOV     (SP)+,R3
005726  012602                          MOV     (SP)+,R2
005730  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
                                ;
005732  010346                  MOVQ:  MOV     R3,-(SP)
005734  010446                         MOV     R4,-(SP)
                                ;
                                ;  Set up R3 to point to queue header but make it look like we are
                                ;  pointing to a queue element.
                                ;
005736  060503                         ADD     R5,R3           ;Point to correct queue head entry
005740  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
                                ;
005744  010400                         MOV     R4,R0           ;Save address of new queue element
005746                                 DISABL                  ;;;** Disable interrupts **
005754  010304                  1$:    MOV     R3,R4           ;;;Remember current queue element address
005756  016303  000000C                MOV     Q.LINK-Q.BLKN(R3),R3;;;Get address of next queue element
005762  001374                         BNE     1$              ;;;Loop till end of list found
005764  010064  000000C                MOV     R0,Q.LINK-Q.BLKN(R4);;;Add new entry to end of list
005770  005060  000000C                CLR     Q.LINK-Q.BLKN(R0);;;Say it is the end of the list
                                ;
                                ;  Finished
                                ;
005774                           9$:    ENABL                   ;** Enable interrupts **
006002  012604                         MOV     (SP)+,R4
006004  012603                         MOV     (SP)+,R3
006006  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.
                                ;
006010  010446                  RTNQ:   MOV     R4,-(SP)
006012  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.
                                ;
006014  005237  000002'                 INC     RTNCNT          ;Is someone else already in this routine?
006020  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.
                                ;
006022                                  DISABL                  ;** Disable interrupts **
006030  005737  000000G                 TST     CLABF           ;;;Is handler currently being held?
006034  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.
                                ;
006036  005737  000006'                 TST     ABTQFL          ;;;Have we already queued a fork request?
006042  001061                          BNE     3$              ;;;Br if yes
006044  005237  000006'                 INC     ABTQFL          ;;;Set flag saying abort fork request queued
006050                                  ENABL                   ;** Enable interrupts **
006056  004737  000000G                 CALL    FRKGET          ;Get a free fork request block
006062  112764  177777G 000000G         MOVB    #<FP$IOA-1>,FQ$PRI(R4);Set priority below I/O abort
006070  012764  006010' 000000G         MOV     #RTNQ,FQ$RTN(R4);Set address of routine to be called by fork
006076  004737  000000G                 CALL    FORKQ           ;Queue the fork request
006102  000441                          BR      3$              ;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.
                                ;
006104  005037  000006'         6$:     CLR     ABTQFL          ;;;Say abort fork request no longer queued
006110  013704  000004'         5$:     MOV     CQH,R4          ;;;Get addr of 1st queue element on compl list
006114  001434                          BEQ     3$              ;;;Br if no more entries to free
006116  016437  000000C 000004'         MOV     Q.LINK-Q.BLKN(R4),CQH ;;;Remove entry from completion list
006124  013746  000000G                 MOV     CLCQE,-(SP)     ;;;Save current queue element pointer
006130  013746  000000G                 MOV     CLLQE,-(SP)     ;;;Also save last queue element pointer
006134  010437  000000G                 MOV     R4,CLCQE        ;;;Set entry being freed as current Q element
006140  010437  000000G                 MOV     R4,CLLQE        ;;;And as last queue element
006144  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
                                ;
006150                          4$:     ENABL                   ;** Enable interrupts **
006156  012704  000000G                 MOV     #CLCQE,R4       ;Point to CQE cell for IOFIN
```

```
     58 006162  004737  000000G                 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 006166                                  DISABL                  ;;;** Disable interrupts **
     64 006174  012637  000000G                 MOV     (SP)+,CLLQE     ;;;Restore saved queue element pointers
     65 006200  012637  000000G                 MOV     (SP)+,CLCQE     ;;;
     66 006204  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 006206  005337  000002'         3$:     DEC     RTNCNT          ;;;Say we are exiting this routine
     71 006212                                  ENABL                   ;** Enable interrupts **
     72                                 ;
     73                                 ;  Finished
     74                                 ;
     75 006220  012605                          MOV     (SP)+,R5
     76 006222  012604                          MOV     (SP)+,R4
     77 006224  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.
                                ;
006226                          LINON:
                                ;
                                ;  Set flag saying line is turned on
                                ;
006226  052765  000000G 000000G         BIS     #CM$ON,CL$STA(R5)       ;Say line is turned on
                                ;
                                ;  Assert Data Terminal Ready
                                ;
006234  052765  000000G 000000G         BIS     #CO$DTR,CL$OPT(R5)      ;Say we want DTR on
006242  004737  006250'                CALL    SETDTR                  ;Raise the DTR line
                                ;
                                ;  Finished
                                ;
006246  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
                                ;
006250  010146                  SETDTR: MOV     R1,-(SP)
006252  010246                          MOV     R2,-(SP)
                                ;
                                ;  See if he wants DTR on or off
                                ;
006254  032765  000000G 000000G         BIT     #CO$DTR,CL$OPT(R5);Is DTR wanted on or off?
006262  001412                          BEQ     1$              ;Br if wanted off
                                ;
                                ;  DTR is wanted on.  See if it is currently on.
                                ;
006264  032765  000000G 000000G         BIT     #CM$DTR,CL$STA(R5)      ;Is DTR currently asserted?
006272  001022                          BNE     9$                      ;Br if yes -- all is ok
006274  052765  000000G 000000G         BIS     #CM$DTR,CL$STA(R5)      ;Say we are raising DTR
006302  012700  000000G                 MOV     #MS$DTR,R0              ;Say we want to set DTR
006306  000410                          BR      2$                      ;Go set DTR
                                ;
                                ;  DTR is wanted off.  See if it is currently off.
                                ;
006310  032765  000000G 000000G 1$:     BIT     #CM$DTR,CL$STA(R5)      ;Is DTR currently off?
006316  001410                          BEQ     9$                      ;Br if yes -- all is ok
006320  042765  000000G 000000G         BIC     #CM$DTR,CL$STA(R5)      ;Say we are dropping DTR
006326  005000                          CLR     R0                      ;Say we want to drop DTR
                                ;
                                ;  Call hardware-dependent routine to change DTR status
                                ;
006330  016501  000000G         2$:     MOV     CL$LIX(R5),R1           ;Get line # for this CL unit
006334  004737  000000G                 CALL    SETDSS                  ;Change DTR status
                                ;
                                ;  Finished
                                ;
006340  012602                  9$:     MOV     (SP)+,R2
006342  012601                          MOV     (SP)+,R1
006344  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$BRK to start sending break; 0 to stop break.
                                ;   R5 = CL unit index number.
                                ;
006346  010146                  SETBRK: MOV     R1,-(SP)
006350  010246                          MOV     R2,-(SP)
                                ;
                                ;  Call hardware-dependent routine to control break transmission
                                ;
006352  016501  000000G                 MOV     CL$LIX(R5),R1   ;Get line # for this CL unit
006356  016102  000000G                 MOV     LCDTYP(R1),R2   ;Get line control type code
006362                                  OCALL   BRKJMP          ;Call hardware control routine
                                ;
                                ;  Finished
                                ;
006370  012602                          MOV     (SP)+,R2
006372  012601                          MOV     (SP)+,R1
006374  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.
                                ;
006376  000172  000000G         BRKJMP: JMP     @CDSBRK(R2)     ;Call hardware routine to control break
        000001                          .END
Errors detected:  0

*** Assembler statistics


Work  file  reads:  0
Work  file writes:  0
Size of work file:  156 Words  ( 1 Pages)
Size of core pool:  18176 Words  ( 71 Pages)
Operating  system:  RT-11

Elapsed time:  00:01:07.04
,LP:TSCLO=DK:TSCLO/C/N:SYM
```

Cross reference table (CREF V05.05)

```
$CTRLS    1-35     9-28    14-14    15-38    30-64
$HISTP    1-39     9-22    14-18    15-59    15-61    30-65
$XCHAR    1-33     8-244
ABTQFL     4-7#   38-30    38-32*   38-44*
BRKJMP    41-17   41-28#
C.CSW      1-41    6-38*   11-26*   12-28*   17-57*
C1DEVX     1-32    5-43
CCICR     18-25   19-21#
CCICTZ    18-38   19-30#
CCILF     18-22   19-15#
CCINUL    18-12   19-9#
CCIRTN    17-80   18-12#
CCISTR    18-13   18-14    18-15    18-16    18-17    18-18    18-19    18-20    18-21    18-23    18-24    18-26
          18-27   18-28    18-29    18-30    18-31    18-32    18-33    18-34    18-35    18-36    18-37    19-4#
          19-10   19-16    19-23
CCOBS     26-18   27-22#
CCOCR     26-23   27-90#
CCOCTL    26-11   26-12    26-13    26-14    26-15    26-16    26-17    26-21    26-24    26-25    26-26    26-27
          26-28   26-29    26-30    26-31    26-32    26-33    26-34    26-35    26-36    26-37    26-38    26-39
          26-40   26-41    27-4#
CCOFF     26-22   27-62    27-75#
CCOLF     26-20   27-47#   27-84
CCONUL    26-10   27-17#   27-39    27-68    27-86
CCORTN   23-114   26-10#
CCOSND    27-6    27-12#   27-24    27-26    27-36    27-38    27-43    27-79    27-93
CCOTAB    26-19   27-30#
CDGDSS     1-55    1-56
CDSBRK     1-57   41-28
CDSDSS     1-55
CDSTRT     1-38   34-17
CDSXON     1-35   14-20    15-63    30-67
CKABTQ     8-207   8-212   35-17    35-19    36-13#
CL$COL     1-47   15-55*   23-48    23-102*  23-105   27-23*   27-25*   27-33*   27-34*   27-37    27-41*   27-91*
CL$EPN     1-37    8-263*   8-348   13-23*   25-18    25-18    25-20*   25-35*
CL$EPP     1-37   13-27*   25-26    25-30*   25-36*
CL$EPS     1-37    8-267    8-356   13-27    25-36
CL$LEN     1-51    8-147*   8-323   23-58    27-48    27-80
CL$LIN     1-51   15-54*   23-58    23-63*   27-47*   27-50    27-52*   27-55    27-61*   27-78*   27-85*
CL$LIX     1-58    5-47     8-239   15-18    17-34    22-19    30-63    34-12    40-35    41-15
CL$OPT     1-45    8-99*    8-115*   8-120*   8-129*   8-138*   8-313   17-74    19-9     19-15    23-80    23-82
          23-92   24-37    27-5     27-31    27-63    27-76    27-92    30-47*   30-55*   32-14*   39-17*   40-15
CL$ORA     1-45    8-237   15-33
CL$ORB     1-49   15-30    23-131   29-78
CL$ORE     1-49   23-129   29-76
CL$ORG     1-31   15-32*
CL$ORP     1-48   15-31*   23-119   23-132*  29-66    29-79*
CL$ORS     1-48    1-49     8-238   15-33*   23-31    23-124*  29-35    29-72*
CL$RQH     1-47    5-65     8-206   17-40    19-21    20-16    21-15    21-30*   35-16
CL$SKP     1-52    8-155*   8-328   27-54
CL$STA     1-45    5-83*    8-35*    8-45*    8-56*    8-91*    8-241    8-318   13-13    13-18*   13-19*   14-28*
          15-42*  15-46    15-50*   17-15    17-22*   17-42    17-54    17-58*   17-101*  19-34*   23-17    23-23*
          23-35   23-46    23-52*   23-56    23-62*   23-72    23-74*   23-87*   23-147*  24-18    25-37*   27-4*
          27-22*  27-30*   27-40*   27-53*   27-57    27-60*   27-65    27-67*   27-75*   27-82*   27-90*   29-18
          29-23*  29-49    29-51*   29-61*   29-95*   30-28*   31-15*   39-13*   40-20    40-22*   40-28    40-30*
CL$WID     1-51    8-163*   8-333   23-103   27-35    27-37
CL$WQH     1-47    5-84     8-211   23-33    24-29    24-60*   35-18
```

Cross reference table (CREF V05.05)

```
CL$XLN    1-32     16-21    22-29    29-29    33-19    33-25*
CLABF     1-47     38-21
CLABRT    1-25     35-9#
CLCLER    1-26      8-20    14-10#
CLCLOS    8-5      13-9#
CLCQE     1-48      5-20     5-22*   38-48    38-50*   38-57    38-65*
CLEOFS    1-37      8-268    8-355
CLERR     5-49      6-15     6-19     6-23     6-25     6-37#
CLGSTS    8-75      8-307    9-12#
CLINCP    1-25     16-10#
CLIOQ     1-25      5-13#
CLLQE     1-48      5-24*   38-49    38-51*   38-64*
CLOCPY   22-31     28-19    29-9#
CLPTWD    8-76      8-178    8-229    8-248    8-308    8-314    8-319    8-324    8-329    8-334    8-344    8-349
         11-15#
CLQOK     5-19#     5-77     5-95    10-16
CLQXIT    5-59      6-39     7-10     7-11     8-6      8-21     8-50     8-81     8-104    8-122    8-131    8-140
          8-148     8-156    8-164    8-183    8-192    8-221    8-230    8-249    8-279    8-286    8-368   10-8#
CLREAD    5-58      5-64#    7-34     8-57
CLREST    1-26      8-95     8-285   15-14#   33-14
CLSFAB    1-39
CLSFBC    1-53
CLSFCH    1-53
CLSFCO    1-54
CLSFDL    1-53
CLSFHS    1-53
CLSFIC    1-45
CLSFMS    1-49
CLSFOC    1-45
CLSFRB    1-53     17-76    17-91
CLSFRL    1-50     19-22
CLSFSL    1-54
CLSFSO    1-54
CLSFSP    1-57
CLSFSS    1-54
CLSFSW    1-54
CLSFWB    1-38
CLSPFN    5-55      6-10#
CLSTRT    8-38      8-46    14-24    23-142   29-90    30-31    30-68    31-16    34-8#
CLTIMR    1-25     22-9#
CLTOTL    1-32     22-15    35-15
CLVERS    1-38      9-16
CLWRIT    5-57      5-82#
CLWRTB    5-83#     7-37
CLXBRK    1-25     29-57    32-10#
CLXDRP   30-75     32-19    33-10#
CLXICP    1-25     28-11#
CLXMCC   29-52     30-13#
CLXSSB   30-34     31-8#
CM$BRK    1-57      8-35     8-45    15-46    15-50    30-28    31-15
CM$CRL    1-56     15-42    23-87    27-4     27-22    27-30    27-65    27-67    27-75    27-90
CM$DTR    1-50     40-20    40-22    40-28    40-30
CM$EFP    1-37      8-241   13-18    15-42    23-35    24-18    25-37
CM$EOF    1-44      8-56    14-28    15-42    17-42    17-54    17-58    19-34
CM$FFI    1-34     23-72    23-74    27-60
CM$FFS    1-51     15-42    23-56    23-62    27-53    27-57    27-82
```

Cross reference table (CREF V05.05)

```
CM$IRG    1-44     17-15    17-22    17-101
CM$MCC    1-34     29-49    29-51    29-61
CM$ON     1-44      8-91    39-13
CM$ORP    1-48     23-17    23-23    23-147   29-18    29-23    29-95
CM$TBS    1-44     15-42    23-46    23-52    27-40
CM$WRT    1-35      5-83    13-13    13-19    15-42
CO$8BT    1-43     23-82
CO$BNI    1-43     17-74    19-9
CO$BNO    1-43     23-80
CO$CR     1-52     27-92
CO$CTL    1-52     27-5
CO$DTR    1-50      8-99     8-115    8-120    30-47    30-55    32-14    39-17    40-15
CO$FF     1-42     27-76
CO$FFO    1-42     24-37
CO$LC     1-51     23-92
CO$LFI    1-42     19-15
CO$LFO    1-42     27-63
CO$TAB    1-42     27-31
CQ$LNK    1-31     30-38*
CQ$LOT    1-30     30-33*
CQ$PA5    1-30     30-35*
CQ$RO     1-31     30-36*
CQ$RTN    1-30     30-34*
CQH       4-6#     10-9     10-10*   21-31    21-32*   24-61    24-62*   36-36    36-37*   38-45    38-47*
CR        3-5#
CS$EOF    1-50     17-57
CS$ERR    1-41      6-38    11-26    12-28
CTRLQ     3-9#
CTRLS     3-8#
CTRLZ     3-10#
EOFCHR   24-24     25-14#
FF        3-7#     23-75    24-40    25-21    27-59
FORCEX    1-36     33-31
FORKQ     1-46     38-37
FP$IOA    1-46     38-35
FQ$PRI    1-46     38-35*
FQ$RTN    1-46     38-36*
FRKGET    1-46     38-34
GETCHR   23-67     24-14#
GETDSS    1-59      8-174     9-34
GETRTQ    1-30     30-32
GETWRD    8-128     8-137     8-146     8-154     8-162     8-189     8-259    12-17#
GTBYT     1-49      8-269    24-51
INPCHR   17-86     19-4     19-24    20-12#
INTPRI    1-33      5-25    10-11    15-34    17-17    17-23    17-50    17-102   21-33    22-37    23-24    23-42
         23-141    23-151   24-63    29-24    29-39    29-89    29-99    30-40    33-26    36-41    36-46    37-32
         38-33     38-56    38-71
IOFIN     1-34     38-58
IRINGG    5-71     16-29    17-9#
KPAR5     1-30     30-35
KPAR6     1-52     12-36*
LCDTYP    1-38      1-55    14-19    15-62    30-66    34-16    41-16
LCLUNT    1-58     16-15
LF        3-6#     23-60    27-69    27-83
LHIRBA    1-40      8-227   15-26    17-44    17-93    29-37
LHIRBB    1-40     15-23
```

Cross reference table (CREF V05.05)

```
LHIRBG    1-40      15-25*
LHIRBP    1-40      15-24*
LHIRBS    1-39       8-228    15-26*    17-44     17-93     29-37
LINIR     1-36      33-21*
LINON     5-64       5-86     39-9#
LNMAP     1-36      33-30
LSW10     1-43       9-22     14-18*    15-59     15-61*    30-65*
LSW3      1-32       1-35      8-244     9-28     14-14*    15-38*    30-64*
LXCL      1-32      28-15     33-20*
MAXSF0    6-16       6-27      7-12#
MAXSF1    6-20       6-26      7-22#
MAXSF2    6-24       7-41#
MOVQ      5-66       5-85     37-11#
MRKTHD    1-31      30-38     30-39*
MS$BRK    1-57       8-36     30-29
MS$CAR    1-56       9-35
MS$DTR    1-55      40-23
MS$RNG    1-56       9-41
NEDCDO    1-39      34-18*
NEDCLO    1-39      34-19*
ORINGP    5-91      13-31     22-33     23-9#
OVRHC     1-55      41-17
PSW       1-33       5-19*     5-25*    10-8*     10-11*    15-22*    15-34*    17-14*    17-17*    17-23*    17-39*    17-50*
         17-102*    21-29*    21-33*    22-37*    23-16*    23-24*    23-30*    23-42*    23-141*   23-151*   24-59*    24-63*
         29-17*     29-24*    29-34*    29-39*    29-89*    29-99*    30-37*    30-40*    33-18*    33-26*    36-25*    36-41*
         36-46*     37-23*    37-32*    38-20*    38-33*    38-56*    38-63*    38-71*
PTBYT     1-38       8-359     8-362     20-22     21-23
PTWRD     1-33      11-34
Q.BLKN    1-34       5-22      5-41      5-43      5-54      5-56*     5-82*     6-37      8-30      8-110     8-202     10-9*
         11-19      11-24     12-21     12-26     12-36     12-40     12-41*    17-56     17-76     17-91     19-22     20-26*
         21-20      21-30     21-31*    24-35     24-35     24-39*    24-39*    24-45     24-50*    24-60     24-61*    36-20
         36-27      36-29     36-35     36-35*    36-36*    37-18     37-25     37-27*    37-28*    38-47     38-52*
Q.BUFF    1-52      11-19     12-21     12-40     12-41*
Q.CSW     1-41       6-37     11-24     12-26     17-56
Q.DEVX    1-30       5-43
Q.FUNC    1-41       5-54     17-76     17-91     19-22
Q.JOB     1-32       8-202    36-29
Q.LINK    1-34       5-22     10-9*     21-30     21-31*    24-60     24-61*    36-20     36-27     36-35     36-35*    36-36*
         37-18      37-25     37-27*    37-28*    38-47     38-52*
Q.PAR     1-52      12-36
Q.UNIT    1-41       5-41
Q.WCNT    1-34       5-56*     5-82*     8-30      8-110     20-26*    21-20     24-45     24-50*
RDFIN    17-59      17-95     19-25     19-38     20-32     21-10#
RTNCNT    4-5#      38-14*    38-70*
RTNQ      8-216     10-12     21-37     24-67     35-25     38-8#     38-36
SETBRK    8-37       8-44     15-49     30-30     31-14     41-10#
SETDSS    1-59      40-36
SETDTR    1-26       8-100     8-121     8-130     8-139     22-24     30-48     30-56     32-15     39-18     40-10#
SETSPD    1-35       8-191
SFABT     7-33       8-201#
SFBREK    7-17       8-30#
SFCLER    7-16       8-14#
SFCLOS    7-8        8-5#
SFCOPT    7-27       8-137#
SFDTR     7-21       8-110#
SFGMS     7-31       8-170#
```

```
SFGOPT    7-40       8-303#
SFGRP0    7-8#       7-12
SFGRP1    7-16#      7-22
SFGRP2    7-26#      7-41
SFIC      7-35       8-227#
SFOC      7-36       8-236#
SFREAD    7-18       8-56#
SFREST    7-39       8-285#
SFSEFP    7-38       8-255#
SFSLEN    7-28       8-146#
SFSOPT    7-26       8-128#
SFSPD     7-32       8-189#
SFSSKP    7-29       8-154#
SFSTAT    7-19       8-73#
SFSWID    7-30       8-162#
SFTERM    7-9        7-20       8-87#
SILFET    1-45      17-64      29-43
SPACE     3-11#     23-50      27-42
SPFRTN    6-33       7-4#
TRNSTR    1-33      16-23
TTINCP    1-36      33-21
VCXCTL    1-31      29-59
VCXTRM    1-31      29-55
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 V05.05)