

Table of contents

3-	1	Parameter definitions
4-	1	Data areas
5-	1	CLENTN -- Entry point for processing a new I/O request
6-	1	CLSPFN -- .SPFUN processing
9-	1	CLGSTS -- Return CL device status
11-	1	CLPTWD -- Store 1 word into user's buffer
12-	1	GETWRD -- Get 1 word from user's buffer
13-	1	CLCLOS -- Initiate end-of-file processing
14-	1	CLCLER -- Clear CL XOFF status
15-	1	CLREST -- Reset a CL unit
16-	1	CLINCP -- Input character processing
17-	1	IRINGG -- Move chars from silo buffer to data buffer
18-	1	CCIRTN -- Input control character processing routines
20-	1	INPCHR -- Move character to user's data buffer
21-	1	RDFIN -- Completed a read request
22-	1	CLTIMR -- Routine called from clock interrupt routine
23-	1	ORINGP -- Move chars from data buffer to output ring buffer
24-	1	GETCHR -- Get next output char from user's data buffer
25-	1	EOFCHR -- Get next end-of-file output character
26-	1	CCORTN -- Output control character processing routines
28-	1	CLXICP -- Got char for output to cross connected CL line
29-	1	CLOCPY -- Copy characters from TT input buf to CL output buf
30-	1	CLXMCC -- Process cross connect modem control character
32-	1	CLXBRK -- Break a CL-TT cross connection and drop DTR
33-	1	CLXDRP -- Break a CL-TT cross connection
34-	1	CLSTRT -- Start transmissions to a line
35-	1	CLABRT -- Handler abort routine
36-	1	CKABTQ -- Check for aborted queue elements
37-	1	MOVQ -- Move queue element to internal queue
38-	1	RTNQ -- Return completed queue elements to the system
39-	1	LINON -- Turn on a communications line
40-	1	SETDTR -- Set Data Terminal Ready status
41-	1	SETBRK -- Control break transmission

```

1          .TITLE  TSCLO -- Communication Line (CL) Handler for TSX-Plus
2 000000    .PSECT  TSCLO
3          .ENABL  LC
4          .ENABL  AMA
5          .DSABL  GBL
6 000000    012257 .RAD50  /CLO/          ;Virtual segment ID word
7          ;-----
8          ; TSCLO is a system virtual overlay which provides support for the
9          ; Communication Line (CL) handler for TSX-Plus.
10         ; This handler supports I/O to communication lines declared by
11         ; use of the IOLINE macro when the system is generated.
12         ; The device names are CLO, CL1, ..., CL7, C10, C11, ..., C17.
13         ; Internal queueing is used to allow concurrent input/output operations
14         ; to take place on all of the devices at the same time.
15         ; XON/XOFF support is provided.
16         ;
17         ; Copyright (c) 1984, 1985.
18         ; S&H Computer Systems, Inc.
19         ; Nashville, Tennessee USA
20         ; All rights reserved
21         ;
22         ;
23         ; Global definitions
24         ;
25         .GLOBL  CLIOQ, CLABRT, CLTIMR, CLINCP, CLXICP, CLXBRK
26         .GLOBL  SETDTR, CLREST, CLCLER
27         ;
28         ; Global references
29         ;
30         .GLOBL  GETRTQ, CQ$LOT, CQ$RTN, KPAR5, CQ$PA5, Q. DEVX
31         .GLOBL  CQ$RO, MRKTHD, CQ$LNK, VCXTRM, VCXCTL, CL$ORG
32         .GLOBL  CLTOTL, LSW3, Q. JOB, LXCL, CL$XLN, C1DEVX
33         .GLOBL  PSW, INTPRI, PTWRD, $XCHAR, TRNSTR
34         .GLOBL  Q. WCNT, Q. BLKN, Q. LINK, IOFIN, CM$MCC, CM$FFI
35         .GLOBL  $CTRLS, LSW3, SETSPD, CDSXON, CM$WRT
36         .GLOBL  TTINCP, LINIR, FORCEX, LNMAP
37         .GLOBL  CL$EPN, CL$EPS, CL$EPP, CM$EFP, CLEDFS
38         .GLOBL  CDSTRT, LCDTYP, PTBYT, CLVERS, CLSFWB
39         .GLOBL  LHIRBS, $HISTP, CLSFAB, NEDCDO, NEDCLO
40         .GLOBL  LHIRBB, LHIRBG, LHIRBP, LHIRBA
41         .GLOBL  Q. UNIT, Q. FUNC, Q. CSW, CS$ERR, C. CSW
42         .GLOBL  CO$FF, CO$TAB, CO$LFO, CO$LFI, CO$FFO
43         .GLOBL  CO$BNO, CO$BNI, LSW10, CO$BBT
44         .GLOBL  CM$TBS, CM$IRG, CM$ON, CM$EOF
45         .GLOBL  CL$OPT, CL$STA, SILFET, CL$ORA, CLSFIC, CLSFOC
46         .GLOBL  FRKGET, FORKQ, FQ$PRI, FQ$RTN, FP$IOA
47         .GLOBL  CL$COL, CL$RQH, CL$WQH, CLABF
48         .GLOBL  CLCQE, CLLQE, CM$ORP, CL$ORS, CL$ORP
49         .GLOBL  CL$ORS, CL$ORE, CL$ORB, GTBYT, CLSFMS
50         .GLOBL  CS$EOF, CO$DTR, CM$DTR, CLSFRL
51         .GLOBL  CM$FFS, CL$LIN, CL$LEN, CO$LC, CL$WID
52         .GLOBL  CO$CTL, CL$SKP, CO$CR, KPAR6, Q. PAR, Q. BUFF
53         .GLOBL  CLSFCH, CLSFBC, CLSFRB, CLSFHS, CLSFDL
54         .GLOBL  CLSFSO, CLSFCD, CLSFSL, CLSFSS, CLSFSW
55         .GLOBL  MS$DTR, CDSOSS, CDQDSS, OVRHC, LCDTYP
56         .GLOBL  CM$CRL, CDQDSS, MS$CAR, MS$RNG
57         .GLOBL  CM$BRK, CDSBRK, MS$BRK, CLSFSP

```

58
59

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

```

1      ;
2      ;-----
3      ; Macro definitions
4      ;
5      ; Disable interrupts
6      ;
7      .MACRO  DISABL          ;Disable interrupts
8      BIS    #340, @PSW
9      .ENDM   DISABL
10     ;
11     ; Enable interrupts
12     ;
13     .MACRO  ENABL          ;Enable interrupts
14     BIC     INTPRI, @PSW
15     .ENDM   ENABL
16     ;
17     ; Call another system virtual overlay region
18     ;
19     .MACRO  OCALL  ENTADD
20     CALL    OVRHC
21     .WORD   ENTADD
22     .ENDM   OCALL

```

Parameter definitions

1				.SBTTL	Parameter definitions	
2				-----		
3				; Ascii characters		
4				;		
5	000015	CR	=	15		; Carriage return
6	000012	LF	=	12		; Line feed
7	000014	FF	=	14		; Form feed
8	000023	CTRLS	=	23		; Ctrl-S
9	000021	CTRLQ	=	21		; Ctrl-Q
10	000032	CTRLZ	=	32		; Ctrl-Z
11	000040	SPACE	=	40		; Space

Data areas

1			.SBTTL Data areas	
2			-----	
3			; General data areas	
4			;	
5	000002	177777	RTNCNT: .WORD -1	;Counts if someone in RTNQ routine
6	000004	000000	CQH: .WORD 0	;List head for Q elements waiting to be freed
7	000006	000000	ABTQFL: .WORD 0	;non-zero ==> RTNQ fork request pending

CLENTR -- Entry point for processing a new I/O request

```

1          .SBTTL  CLENTR -- Entry point for processing a new I/O request
2          ;-----
3          ; CLIOQ is called by the system I/O initiation routine to start a new
4          ; I/O request.
5          ; We process some requests immediately, but for most (such as read and
6          ; write) we move the request from the handler queue onto an internal
7          ; queue.
8          ;
9          ; Inputs:
10         ;   CLCQE = Current queue request.
11         ;   CLLQE = Last queue request.
12         ;
13 000010   010346  CLIOQ:  MOV      R3,-(SP)
14 000012   010446      MOV      R4,-(SP)
15 000014   010546      MOV      R5,-(SP)
16         ;
17         ; Remove current queue element from list pointed to by handler header
18         ;
19 000016         CLQOK:  DISABL                    ;** Disable interrupts **
20 000024   013704   000000G      MOV      CLCQE,R4      ;Get pointer to queue element
21 000030   001406      BEQ      1$                    ;Br if there is no queue element to process
22 000032   016437   000000C 000000G      MOV      Q.LINK-Q.BLKN(R4),CLCQE ;Remove queue element from list
23 000040   001002      BNE      1$                    ;Br if more elements pending
24 000042   005037   000000G      CLR      CLLQE        ;Say there are no pending queue elements
25 000046         1$:  ENABL                    ;** Enable interrupts **
26 000054   005704      TST      R4                    ;Is there a queue element to process?
27 000056   001004      BNE      3$                    ;Br if yes
28         ;
29         ; There are no remaining queue elements for the handler to process.
30         ; Return to the system.
31         ;
32 000060   012605      MOV      (SP)+,R5
33 000062   012604      MOV      (SP)+,R4
34 000064   012603      MOV      (SP)+,R3
35 000066   000207      RETURN
36         ;
37         ; There is a queue request to be processed.
38         ; R4 = Points to Q.BLKN cell in queue element.
39         ; Determine if I/O is being done to a valid CL unit
40         ;
41 000070   116405   000000C 3$:  MOVB     Q.UNIT-Q.BLKN(R4),R5 ;Get device unit number
42 000074   042705   177770      BIC      #^C7,R5      ;Clear all but unit # field
43 000100   126437   000000C 000000G      CMPB     Q.DEVX-Q.BLKN(R4),C1DEVX ;Is the a C1 unit?
44 000106   001002      BNE      4$                    ;Br if not
45 000110   062705   000010      ADD      #8.,R5        ;Bias C1 unit numbers by 8
46 000114   006305 4$:  ASL      R5                    ;Convert to word index
47 000116   016501   000000G      MOV      CL$LIX(R5),R1 ;Is this CL unit associated with a line?
48 000122   001002      BNE      2$                    ;Br if yes -- This is a valid CL unit
49 000124   000137   000322'      JMP      CLERR        ;Return immediate hard error code
50         ;
51         ; Get the function code and see if this is a .READ, .WRITE, or .SPFUN.
52         ; R5 = CL unit index number.
53         ;
54 000130   116403   000000C 2$:  MOVB     Q.FUNC-Q.BLKN(R4),R3 ;Get the function code
55 000134   001037      BNE      CLSPFN                ;Br if this is a .SPFUN operation
56 000136   006364   000000C      ASL      Q.WCNT-Q.BLKN(R4) ;Convert word count to # bytes
57 000142   103415      BCS      CLWRIT                ;Br if this is a write operation

```

CLENTR -- Entry point for processing a new I/O request

```

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$RQH,R3      ;Get pointer to read queue head
66 000162 004737 005732'   CALL      MOVQ      ;Move queue element to internal queue
67
68                          ; Call routine to move any pending characters in silo buffer for this
69                          ; line into the data buffer.
70
71 000166 004737 002212'   CALL      IRINGG      ;Move chars from silo buffer to data buffer
72
73                          ; Finished starting the read operation.
74                          ; The queue element will be returned to the system when we have
75                          ; completed the operation.
76
77 000172 000137 000016'   JMP      CLQOK
78
79                          ; This is a .WRITE request.
80                          ; Move queue element to internal write queue for this CL unit.
81
82 000176 005464 000000C   CLWRIT: NEG      Q.WCNT-Q.BLKN(R4) ;Make write byte count positive
83 000202 052765 000000G 000000G CLWRTB: BIS      #CM$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

```

CLSPFN -- .SPFUN processing

```

1          .SBTTL  CLSPFN -- .SPFUN processing
2          -----
3          ; The current queue request is for a .SPFUN operation
4          ; At this point the following registers are set up:
5          ;   R1 = TSX-Plus line index number of line being used by CL unit.
6          ;   R3 = .SPFUN code from Q.FUNC.
7          ;   R4 = Pointer to Q.BLKN field of current queue element
8          ;   R5 = CL unit index number
9          ;
10         CLSPFN:
11         ;
12         ; See which group of special functions this code is in
13         ;
14         000234 042703 177400      BIC      #^C<377>,R3      ;Clear sign extension
15         000240 001430              BEQ      CLERR           ;Function code of 0 is invalid
16         000242 020327 000004      CMP      R3,#MAXSF0      ;Too big for group 0?
17         000246 101420              BLOS     3$             ;Br if in group 0
18         000250 020327 000201      CMP      R3,#201          ;Is this code too small?
19         000254 103422              BLO      CLERR           ;Br if too small
20         000256 020327 000206      CMP      R3,#MAXSF1        ;Is it in group 1?
21         000262 101410              BLOS     2$             ;Br if yes
22         000264 020327 000250      CMP      R3,#250          ;Is it in group 2?
23         000270 103414              BLO      CLERR           ;Br if too small for group 2
24         000272 020327 000266      CMP      R3,#MAXSF2        ;Is it within group 2?
25         000276 101011              BHI      CLERR           ;Br if not
26         000300 162703 000041      SUB      #247-MAXSF1,R3    ;Correct for group 1 codes
27         000304 162703 000174      2$:      SUB      #200-MAXSF0,R3 ;Correct for group 0 codes
28         ;
29         ; Branch off to processing routine
30         ;
31         000310 162703 000001      3$:      SUB      #1,R3      ;Subtract lowest function code
32         000314 006303              ASL      R3              ;Convert function code to word table index
33         000316 000173 000340'      JMP      @SPFRTN(R3)      ;Enter processing routine
34         ;
35         ; Invalid special function code
36         ;
37         000322 016400 000000C      CLERR:  MOV      Q.CSW-Q.BLKN(R4),R0 ;Get address of CSW
38         000326 052760 000000G 000000G  BIS      #CS$ERR,C.CSW(R0) ;Set hard error flag in CSW
39         000334 000137 001502'      JMP      CLQXIT          ;Do .DRFIN to tell system this op is completed

```

CLSPFN -- .SPFUN processing

```

1          ; -----
2          ;   Branch vector for .SPFUN processing routines based on function code value.
3          ;
4 000340   SPFRTN:
5          ;
6          ;   Group 0: Function codes in the range 1 to 4
7          ;
8 000340   SFGRP0: .WORD   SFCLOS           ;001 - Close file
9 000342   .WORD   SFTERM           ;002 - Delete file
10 000344   .WORD   CLQXIT          ;003 - Lookup file
11 000346   .WORD   CLQXIT          ;004 - Enter file
12         000004   MAXSFO = <.-SFGRP0>/2 ;Maximum function code value in group 0
13         ;
14         ;   Group 1: Function codes in the range 201 to 247.
15         ;
16 000350   SFGRP1: .WORD   SFCLER           ;201 - Clear flags
17 000352   .WORD   SFBREK          ;202 - Break transmission control
18 000354   .WORD   SFREAD          ;203 - Special read with byte count
19 000356   .WORD   SFSTAT          ;204 - Get handler status
20 000360   .WORD   SFTERM          ;205 - Terminate I/O
21 000362   .WORD   SFDTR           ;206 - Raise or drop DTR signal
22         000206   MAXSF1 = 200+<<.-SFGRP1>/2> ;Highest function code in group 1
23         ;
24         ;   Group 2: Function codes with values of 250 or greater.
25         ;
26 000364   SFGRP2: .WORD   SFSOPT          ;250 - Set option flags
27 000366   .WORD   SFCOPT          ;251 - Clear option flags
28 000370   .WORD   SFSLEN          ;252 - Set page length
29 000372   .WORD   SFSSKP          ;253 - Set skip lines
30 000374   .WORD   SFSWID          ;254 - Set page width
31 000376   .WORD   SFGMS           ;255 - Get modem status
32 000400   .WORD   SFSPD           ;256 - Set transmit/receive speed
33 000402   .WORD   SFABT           ;257 - Abort all pending read/write requests
34 000404   .WORD   CLREAD          ;260 - Read line with byte count
35 000406   .WORD   SFIC            ;261 - Get number of pending input characters
36 000410   .WORD   SFOC            ;262 - Get number of pending output chars
37 000412   .WORD   CLWRTB          ;263 - Write with byte count
38 000414   .WORD   SFSEFP          ;264 - Set end-of-file output control
39 000416   .WORD   SFREST          ;265 - Reset CL unit
40 000420   .WORD   SFGOPT          ;266 - Get current options and settings
41         000266   MAXSF2 = 247+<<.-SFGRP2>/2> ;Highest legal function # in group 2

```

```

1      ;-----
2      ; Special function # 1
3      ; Close file.
4      ;
5 000422 004737 001660' SFCLOS: CALL    CLCLOS      ;Perform end-of-file operations
6 000426 000137 001502'      JMP      CLQXIT      ;Finished
7      ;
8      ;-----
9      ; Special function # 201
10     ; Clear handler flags.
11     ; The effect is to clear the flag saying we have received an XOFF
12     ; and to send an XON.
13     ;
14 000432 SFCLER:
15     ;
16     ; R1 contains index of TS line in use as CL unit
17     ; R5 contains CL unit index
18     ; Call the routine shared with the equivalent EMT to do the work
19     ;
20 000432 004737 001724'      CALL    CLCLER      ;Go do it
21 000436 000137 001502'      JMP      CLQXIT      ;Finished with operation
22     ;
23     ;-----
24     ; Special function # 202
25     ; Start or stop sending a break.
26     ; Word count non-zero ==> Start sending a break.
27     ; Word count zero      ==> End sending a break.
28     ;
29     ;
30 000442 005764 000000C SFBREK: TST      Q.WCNT-Q.BLKN(R4) ;Start or end break?
31 000446 001412      BEQ      1$              ;Br if we are ending a break
32     ;
33     ; Begin sending a break
34     ;
35 000450 052765 000000G 000000G      BIS      #CM$BRK,CL$STA(R5);Set flag saying we are sending a break
36 000456 012700 000000G      MOV      #MS$BRK,R0      ;Set flag to start break transmission
37 000462 004737 006346'      CALL    SETBRK      ;Call hardware routine to start sending break
38 000466 004737 005514'      CALL    CLSTRT      ;Start transmitter
39 000472 000410      BR      9$
40     ;
41     ; End sending a break
42     ;
43 000474 005000 1$:      CLR      R0              ;Clear break-send flag
44 000476 004737 006346'      CALL    SETBRK      ;Call hardware routine to end break
45 000502 042765 000000G 000000G      BIC      #CM$BRK,CL$STA(R5);Clear flag that says we are sending a break
46 000510 004737 005514'      CALL    CLSTRT      ;Start transmitter
47     ;
48     ; Finished
49     ;
50 000514 000137 001502' 9$:      JMP      CLQXIT      ;Finished with .SPFUN
51     ;
52     ;-----
53     ; Special function # 203
54     ; Read with Q.WCNT indicating the byte count rather than the word count
55     ;
56 000520 042765 000000G 000000G SFREAD: BIC      #CM$EOF,CL$STA(R5);Clear end of file status
57 000526 000137 000152'      JMP      CLREAD      ;Enter read routine (Q.WCNT = byte count)

```

CLSPFN -- .SPFUN processing

```

58 ;
59 ;-----
60 ; Special function # 204
61 ; Get handler status.
62 ; The following information is stored into the first word of the
63 ; user's buffer:
64 ;   High order byte: Handler version number
65 ;   Low order byte:
66 ; XL$XFX bit 0: 1 ==> We have sent XOFF to stop transmission to us.
67 ; XL$XFR bit 1: 1 ==> We have received an XOFF.
68 ; XL$CTS bit 2: 1 ==> Clear To Send (CTS) is asserted.
69 ; Next two are RT 5.4 compatible
70 ; XL$CD  bit 3: 1 ==> Carrier is detected
71 ; XL$RI  bit 4: 1 ==> Ring is detected
72 ;
73 000532 010246 SFSTAT: MOV      R2,-(SP)
74 ;
75 000534 004737 001400'      CALL      CLGSTS      ;Call common routine to get status
76 000540 004737 001540'      CALL      CLPTWD      ;Store value into user's buffer
77 ;
78 ; Finished
79 ;
80 000544 012602      MOV      (SP)+,R2
81 000546 000137 001502'      JMP       CLQXIT      ;Finished with operation
82 ;
83 ;-----
84 ; Special function # 205
85 ; Terminate I/O to the line.
86 ;
87 000552 SFTERM:
88 ;
89 ; Set flag saying to ignore input from the line
90 ;
91 000552 042765 000000G 000000G      BIC      #CM$ON,CL$STA(R5); Say line is turned off
92 ;
93 ; Clear input and output silos and other CL unit status
94 ;
95 000560 004737 001764'      CALL      CLREST      ;Reset the CL unit
96 ;
97 ; Drop Data Terminal Ready
98 ;
99 000564 042765 000000G 000000G      BIC      #CO$DTR,CL$OPT(R5) ; Say we want DTR off
100 000572 004737 006250'      CALL      SETDTR      ;Call routine to drop DTR
101 ;
102 ; Finished
103 ;
104 000576 000137 001502'      JMP       CLQXIT
105 ;
106 ;-----
107 ; Special function # 206
108 ; Raise or drop DTR signal.
109 ;
110 000602 005764 000000C      SFDTR:  TST      @.WCNT-Q.BLKN(R4) ; Raise or drop DTR?
111 000606 001004      BNE      1$          ;Br if raising DTR
112 ;
113 ; Drop DTR
114 ;

```

```

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      RO,CL$OPT(R5) ;Set specified option flags
130 000646 004737 006250'      CALL      SETDTR      ;Check for DTR status change
131 000652 000137 001502'      JMP      CLQXIT
132                                     ;
133                                     ; -----
134                                     ; Special function # 251
135                                     ; Clear option flags
136                                     ;
137 000656 004737 001602'      SFCOPT: CALL      GETWRD      ;Get word from user's buffer
138 000662 040065 000000G      BIC      RO,CL$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      RO,CL$LEN(R5) ;Set page length for this unit
148 000706 000137 001502'      JMP      CLQXIT
149                                     ;
150                                     ; -----
151                                     ; Special function # 253
152                                     ; Set number of lines to skip at bottom of page.
153                                     ;
154 000712 004737 001602'      SFSSKP: CALL      GETWRD      ;Get word from user's buffer
155 000716 010065 000000G      MOV      RO,CL$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      RO,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                                     ;

```

```

172      ; Call hardware dependent routine to get the modem status
173
174 000744 004737 000000G      ;      CALL      GETDSS      ;Call routine to get the data set status
175
176      ; Return status value to 1st word of user's buffer
177
178 000750 004737 001540'      ;      CALL      CLPTWD      ;Store value into 1st word of user's buffer
179
180      ; Finished
181
182 000754 012602      MOV      (SP)+,R2
183 000756 000137 001502'      JMP      CLQXIT      ;Finished I/O operation
184
185      ;-----
186      ; Special function # 256.
187      ; Set transmit/receive speed.
188
189 000762 004737 001602' SFSPD: CALL      GETWRD      ;Get word from user's buffer
190 000766 103402      BCS      1$      ;Br if invalid buffer address
191 000770 004737 000000G      CALL      SETSPD      ;Set the speed
192 000774 000137 001502' 1$:      JMP      CLQXIT      ;Finished
193
194      ;-----
195      ; Special function # 257.
196      ; Abort all pending read and write requests for the job.
197
198      ; Inputs:
199      ; R4 = Pointer to 3rd word of .SPFUN queue element.
200
201 001000 010446 SFABT: MOV      R4,-(SP)      ;Save pointer to current queue element
202 001002 116404      MOVB      Q.JOB-Q.BLKN(R4),R4 ;Get job # from .SPFUN queue element
203
204      ; Abort pending read requests for this job
205
206 001006 012703      MOV      #CL$RQH,R3      ;Point to read queue head
207 001012 004737 005622'      CALL      CKABTQ      ;Abort pending reads for job
208
209      ; Abort pending write requests for this job
210
211 001016 012703      MOV      #CL$WQH,R3      ;Point to write queue head
212 001022 004737 005622'      CALL      CKABTQ      ;Abort pending writs for job
213
214      ; Call routine to return any freed queue elements to the system
215
216 001026 004737 006010'      CALL      RTNQ      ;Return freed queue elements to the system
217
218      ; Finished
219
220 001032 012604      MOV      (SP)+,R4      ;Restore pointer to queue element
221 001034 000137 001502'      JMP      CLQXIT      ;Finished operation
222
223      ;-----
224      ; Special function # 261.
225      ; Get number of bytes pending in input silo buffer.
226
227 001040 016100 SFIC: MOV      LHIRBA(R1),R0      ;Get allocated size of input buffer
228 001044 166100      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          SFDC:    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

```

```

286 001220 000137 001502'      JMP      CLQXIT      ;Finished
287
288
289      ;-----
290      ; Special function # 266
291      ; Get current options and settings.
292
293      ; Returns 13 words to user buffer
294      ; 1  Handler status as for SPFUN 204
295      ; 2  CL options flags
296      ; 3  internal flags word
297      ; 4  page length
298      ; 5  end of page skip lines
299      ; 6  page width
300      ; 7  TS line number and CL unit number
301      ; 8  number end of file form feeds
302      ; 9-12 end of file string, ASCIZ up to CLEOFS long
303 001224 SFGOPT:
304
305      ; Get and return status word just as for SPFUN 204
306      ; word 1
307 001224 004737 001400'      CALL     CLGSTS      ;Get current status word
308 001230 004737 001540'      CALL     CLPTWD      ;Return to user buffer
309 001234 103457      BCS      9%      ;Error return if no chan or odd buff addr
310
311      ; Return CL options word
312      ; word 2
313 001236 016500 000000G      MOV      CL$OPT(R5),R0 ;Get options word
314 001242 004737 001540'      CALL     CLPTWD      ;Return to user buffer
315
316      ; Return internal status word
317      ; word 3
318 001246 016500 000000G      MOV      CL$STA(R5),R0 ;Get internal status word
319 001252 004737 001540'      CALL     CLPTWD      ;Return to user buffer
320
321      ; Return current page length
322      ; word 4
323 001256 016500 000000G      MOV      CL$LEN(R5),R0 ;Get current length
324 001262 004737 001540'      CALL     CLPTWD      ;Return to user buffer
325
326      ; Return current number lines to skip at end of page
327      ; word 5
328 001266 016500 000000G      MOV      CL$SKP(R5),R0 ;Get current # skip lines
329 001272 004737 001540'      CALL     CLPTWD      ;Return to user buffer
330
331      ; Return current page width
332      ; word 6
333 001276 016500 000000G      MOV      CL$WID(R5),R0 ;Get current width
334 001302 004737 001540'      CALL     CLPTWD      ;Return to user buffer
335
336      ; Line number being used as CL unit --> low byte
337      ; CL unit number --> high byte (will be > 7 if C1 unit);
338      ; word 7
339 001306 010146      MOV      R1,-(SP)      ;Get T/S line index
340 001310 010500      MOV      R5,R0      ;Get CL unit index
341 001312 000300      SWAB     R0      ;Move to high byte
342 001314 052600      BIS      (SP)+,R0      ;Merge in line index

```

CLSPFN -- .SPFUN processing

```

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$EPN(R5),R0  ;Get EOF FF's
349 001330 004737 001540'  CALL     CLPTWD      ;Return to user buffer
350                               ;
351       ; Return current end of file string
352       ; words 9 - 12
353 001334 010246          MOV      R2,-(SP)      ;Save registers
354 001336 010346          MOV      R3,-(SP)
355 001340 012703 000000G   MOV      #CLEDFS,R3    ;Get number of chars to move
356 001344 016502 000000G   MOV      CL$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

```

```

1          .SBTTL  CLGSTS -- Return CL device status
2          -----
3          ; CLGSTS is called by CL .SPFUNs 204 and 266 to return the CL version
4          ; number and modem status bits in R0.
5          ; Inputs:
6          ;     R1 index number of line being used as CL unit
7          ;     R5 contains the CL unit index number
8          ; Outputs:
9          ;     R0 contains the version and status bits
10         ;     (see .SPFUN 204 for complete bit description)
11
12 001400 010346 CLGSTS: MOV      R3,-(SP)      ; Save R3
13
14          ; Get version number to high-order byte
15
16 001402 113703 000000G      MOVB      CLVERS,R3      ; Get version number
17 001406 042703 177400      BIC        #^C377,R3      ; Kill possible sign extension
18 001412 000303              SWAB      R3              ; Move version to high byte
19
20          ; See if we have sent an XOFF to stop transmission to us
21
22 001414 032761 000000G 000000G      BIT        ##HISTP,LSW10(R1); Have we send XOFF?
23 001422 001402              BEQ        1$              ; Br if not
24 001424 052703 000000G      BIS        #XL$XFX,R3      ; Set status flag
25
26          ; See if we have received an XOFF
27
28 001430 032761 000000G 000000G 1$: BIT        ##CTRLS,LSW3(R1); Have we received an XOFF?
29 001436 001402              BEQ        2$              ; Br if not
30 001440 052703 000000G      BIS        #XL$XFR,R3      ; Set status flag
31
32          ; See if Clear To Send (CTS) is asserted
33
34 001444 004737 000000G      2$: CALL      GETDSS      ; Call routine to get dataset status
35 001450 032700 000000G      BIT        #MS$CAR,R0      ; Is carrier detected?
36 001454 001402              BEQ        3$              ; Br if not
37 001456 052703 000000C      BIS        #<XL$CTS!XL$CD>,R3 ; Say CTS is asserted and ring detected
38
39          ; See if Ring is asserted
40
41 001462 032700 000000G      3$: BIT        #MS$RNG,R0      ; Is ring detected?
42 001466 001402              BEQ        4$              ; Br if not
43 001470 052703 000000G      BIS        #XL$RI,R3      ; Say ring is detected
44
45          ; Return status value in R0
46
47 001474 010300      4$: MOV      R3,R0      ; Get value to R0 for CLPTWD
48 001476 012603      MOV      (SP)+,R3      ; Restore R3
49 001500 000207      RETURN

```

CLGSTS -- Return CL device status

```

1      ; -----
2      ; We completed the I/O operation.
3      ; Return the queue element to the system.
4      ;
5      ; Inputs:
6      ;   R4 = Address of current queue element.
7      ;
8 001502 CLQXIT: DISABL      ;;;** Disable interrupts **
9 001510 013764 000004' 000000C      MOV    CQH,Q.LINK-Q.BLKN(R4);; Put queue element on completed list
10 001516 010437 000004'      MOV    R4,CQH      ;;;
11 001522      ENABL      ;;;** Enable interrupts
12 001530 004737 006010'      CALL    RTNQ      ;Return queue element to the system
13      ;
14      ; Go back and see if there is another queue element pending
15      ;
16 001534 000137 000016'      JMP     CLQOK      ;Go back and check for another request

```

CLPTWD -- Store 1 word into user's buffer

```

1          .SBTTL  CLPTWD -- Store 1 word into user's buffer
2          ;-----
3          ; CLPTWD is called from some of the .SPFUN processing routines to store
4          ; a one word value into the 1st word of the user's data buffer.
5          ; If the buffer address is odd, the error flag is set in the channel
6          ; status word, the C-flag is set on return, and the value is not stored.
7          ;
8          ; Inputs:
9          ;   R0 = Value to store.
10         ;   R4 = Pointer to current queue element.
11         ;
12         ; Outputs:
13         ;   C-flag set ==> Error: buffer address odd
14         ;
15 001540   CLPTWD:
16         ;
17         ; See if the buffer address is odd
18         ;
19 001540   032764   000001   000000C      BIT      #1,Q.BUFF-Q.BLKN(R4) ;Is the buffer address odd?
20 001546   001410                      BEQ      1$          ;Br if not
21         ;
22         ; Error: The buffer address is odd
23         ;
24 001550   016400   000000C      MOV      Q.CSW-Q.BLKN(R4),R0 ;Get address of CSW for channel
25 001554   001403                      BEQ      2$          ;Br if no channel address
26 001556   052760   000000G 000000G      BIS      #CS$ERR,C.CSW(R0);Set error flag in CSW
27 001564   000261   2$:          SEC                      ;Signal error on return
28 001566   000404                      BR      9$
29         ;
30         ; Buffer address is OK.
31         ; Call PTWRD to store the value.
32         ;
33 001570   010046   1$:          MOV      R0,-(SP)          ;Stack the value for PTWRD
34 001572   004737   000000G      CALL     PTWRD           ;Store value into user's buffer
35 001576   000241                      CLC              ;Signal success on return
36         ;
37         ; Finished
38         ;
39 001600   000207   9$:          RETURN

```

GETWRD -- Get 1 word from user's buffer

```

1          .SBTTL  GETWRD -- Get 1 word from user's buffer
2          ;-----
3          ; GETWRD is called from some of the .SPFUN processing routines to get
4          ; a one word value from the 1st word of the user's data buffer.
5          ; If the buffer address is odd, the error flag is set in the channel
6          ; status word, the C-flag is set on return, and 0 (zero) is returned
7          ; in R0.
8          ;
9          ; Inputs:
10         ;   R4 = Pointer to current queue element
11         ;
12         ; Outputs:
13         ;   R0 = Value from 1st word of data buffer
14         ;   C-flag set ==> Buffer address was odd (R0 contains 0 in this case).
15         ;   Buffer address is incremented by 2 in queue element.
16         ;
17 001602   GETWRD:
18         ;
19         ; See if the buffer address is odd
20         ;
21 001602   032764   000001   000000C       BIT      #1,Q.BUFF-Q.BLKN(R4)   ;Is the buffer address odd?
22 001610   001411               BEQ      1$                               ;Br if not odd
23         ;
24         ; Error: The buffer address is odd
25         ;
26 001612   016400   000000C               MOV      Q.CSW-Q.BLKN(R4),R0       ;Get address of channel status word
27 001616   001403               BEQ      2$                               ;Br if there is none
28 001620   052760   000000G 000000G       BIS      #CS$ERR,C.CSW(R0)       ;Set error flag in channel status
29 001626   005000   2$:          CLR      R0                               ;Return 0 in R0
30 001630   000261               SEC                               ;Signal error on return
31 001632   000411               BR       9$
32         ;
33         ; Buffer address is ok.
34         ; Map PAR6 to user's buffer.
35         ;
36 001634   016437   000000C 000000G 1$:    MOV      Q.PAR-Q.BLKN(R4),@#KPAR6 ;Map KPAR6 to user's buffer
37         ;
38         ; Get word from the buffer
39         ;
40 001642   017400   000000C               MOV      @Q.BUFF-Q.BLKN(R4),R0     ;Get value from buffer
41 001646   062764   000002   000000C       ADD      #2,Q.BUFF-Q.BLKN(R4)    ;Advance buffer address
42 001654   000241               CLC                               ;Signal success on return
43         ;
44         ; Finished
45         ;
46 001656   000207   9$:          RETURN

```

CLCLOS -- Initiate end-of-file processing

```

1          .SBTTL  CLCLOS -- Initiate end-of-file processing
2          ;-----
3          ; CLCLOS is called when end of file is reached on output processing
4          ; and we want to initiate the end-of-file output processing.
5          ;
6          ; Inputs:
7          ;   R5 = CL unit index
8          ;
9 001660    CLCLOS:
10         ;
11         ; Only do output EOF processing if a write was done to this unit
12         ;
13 001660    032765 000000G 000000G      BIT      #CM$WRT,CL$STA(R5);Was a write done to this unit?
14 001666    001415      BEQ      9$          ;Br if not
15         ;
16         ; Say we are doing end-of-file processing
17         ;
18 001670    052765 000000G 000000G      BIS      #CM$EFP,CL$STA(R5);We have started EOF processing for unit
19 001676    042765 000000G 000000G      BIC      #CM$WRT,CL$STA(R5);Clear write-done flag for unit
20         ;
21         ; Reset form-feed count
22         ;
23 001704    105065 000001G      CLR      CL$EPN+1(R5) ;Say no form-feeds sent yet
24         ;
25         ; Reset ENDSTRING pointer
26         ;
27 001710    016565 000000G 000000G      MOV      CL$EPS(R5),CL$EPP(R5);Reset endstring pointer
28         ;
29         ; Initiate output to the unit
30         ;
31 001716    004737 003130'      CALL      DRINGP          ;Initiate output to unit
32         ;
33         ; Finished
34         ;
35 001722    000207      9$:      RETURN

```

CLCLER -- Clear CL XOFF status

```

1          .SBTTL  CLCLER -- Clear CL XOFF status
2          ;-----
3          ; This routine clears the flag saying that the line has received an
4          ; XOFF and transmits an XON
5          ;
6          ; Inputs:
7          ;   R1      Index of the TS line to which the CL unit is connected
8          ;   R5      CL unit index
9          ;
10         CLCLER:
11         ;
12         ; Clear flag saying we have received an XOFF
13         ;
14         001724  042761  000000G 000000G      BIC      #$CTRLS,LSW3(R1);Clear the ctrl-S flag
15         ;
16         ; Send an XON
17         ;
18         001732  042761  000000G 000000G      BIC      #$HISTP,LSW10(R1);Say input has not been stopped by XOFF
19         001740  016100  000000G      MOV      LCDTYP(R1),R0      ;Get device type code
20         001744  004770  000000G      CALL     @CDSXON(R0)      ;Call routine to stuff XON into output
21         ;
22         ; Start output
23         ;
24         001750  004737  005514'      CALL     CLSTRT      ;Start transmission
25         ;
26         ; Clear end of file flag
27         ;
28         001754  042765  000000G 000000G      BIC      #CM$EOF,CL$STA(R5);Clear end of file status
29         001762  000207      RETURN

```

CLREST -- Reset a CL unit

```

1          .SBTTL  CLREST -- Reset a CL unit
2          -----
3          ; Reset a CL unit.  This consists of the following actions:
4          ;   1. Empty input silo.
5          ;   2. Empty output silo.
6          ;   3. Reset line and column positions.
7          ;   4. Stop sending break if we are currently sending it.
8          ;   5. Clear flag that says we have received an XOFF.
9          ;   6. Send an XON if we previously sent an XOFF.
10         ;
11         ; Inputs:
12         ;   R5 = CL unit number index
13         ;
14 001764 010146 CLREST: MOV      R1,-(SP)
15         ;
16         ; Get line # CL unit is connected to
17         ;
18 001766 016501 000000G      MOV      CL$LIX(R5),R1  ;Get line index number
19         ;
20         ; Clear out the input silo buffer
21         ;
22 001772      DISABL      ;;;** Disable interrupts **
23 002000 016100 000000G      MOV      LHIRBB(R1),RO  ;;;Get pointer to start of silo buffer
24 002004 010061 000000G      MOV      RO,LHIRBP(R1)  ;;;Reset input interrupt pointer
25 002010 010061 000000G      MOV      RO,LHIRBG(R1)  ;;;Reset next available char pointer
26 002014 016161 000000G 000000G      MOV      LHIRBA(R1),LHIRBS(R1);;;Reset free space counter
27         ;
28         ; Clear out the output silo buffer
29         ;
30 002022 016500 000000G      MOV      CL$ORB(R5),RO  ;;;Get pointer to start of output silo buffer
31 002026 010065 000000G      MOV      RO,CL$ORP(R5)  ;;;Output character pointer
32 002032 010065 000000G      MOV      RO,CL$ORG(R5)  ;;;Next available output character
33 002036 016565 000000G 000000G      MOV      CL$ORA(R5),CL$ORS(R5);;;Available space in output buffer
34 002044      ENABL      ;;;** Enable interrupts **
35         ;
36         ; Clear flag that says we have received an XOFF
37         ;
38 002052 042761 000000G 000000G      BIC      #$CTRLS,LSW3(R1);Say line output not suspended due to XOFF
39         ;
40         ; Clear some status flags for the unit
41         ;
42 002060 042765 000000C 000000G      BIC      <CM$WRT!CM$EFP!CM$CRL!CM$TBS!CM$EOF!CM$FFS>,CL$STA(R5)
43         ;
44         ; If we are sending a break, stop now
45         ;
46 002066 032765 000000G 000000G      BIT      #CM$BRK,CL$STA(R5);Are we sending a break now?
47 002074 001406      BEQ      1$          ;Br if not
48 002076 005000      CLR      RO          ;Say to reset break transmission
49 002100 004737 006346'      CALL     SETBRK      ;Stop sending break
50 002104 042765 000000G 000000G      BIC      #CM$BRK,CL$STA(R5);Say break transmission finished
51         ;
52         ; Reset page and line position
53         ;
54 002112 005065 000000G      1$:  CLR      CL$LIN(R5)      ;Say we are at the top of a page
55 002116 005065 000000G      CLR      CL$COL(R5)      ;Say we are at left-most column of line
56         ;
57         ; If we previously sent an XOFF to stop the sender, send an XON now.

```

CLREST -- Reset a CL unit

```
58
59 002122 032761 000000G 000000G      BIT    ##HISTP,LSW10(R1);Did we send an XOFF?
60 002130 001407                      BEQ     9$      ;Br if not
61 002132 042761 000000G 000000G      BIC     ##HISTP,LSW10(R1);Can XOFF has been cleared
62 002140 016100 000000G              MOV     LCDTYP(R1),R0 ;Get line type index
63 002144 004770 000000G              CALL    @CDSXON(R0)    ;Send XON
64
65                      ; Finished
66                      ;
67 002150 012601          9$:      MOV     (SP)+,R1
68 002152 000207                      RETURN
```

CLINCP -- Input character processing

```

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         CLINCP: MOV      R1,-(SP)
11         MOV      R5,-(SP)
12         ;
13         ; Convert line index number to CL unit index
14         ;
15         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         MOV      CL$XLN(R5),R1 ; Is this CL unit cross-connected to TT line?
22         BEQ      1$           ; Br if not
23         CALL     @TRNSTR      ; Try to start output to TT line
24         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         1$: CALL     IRINGG      ; Move chars to user's data buffer
30         ;
31         ; Finished
32         ;
33         9$: MOV      (SP)+,R5
34         MOV      (SP)+,R1
35         RETURN

```

IRINGG -- Move chars from silo buffer to data buffer

```

1          .SBTTL  IRINGG -- Move chars from silo buffer to data buffer
2          -----
3          ; IRINGG is called to move all characters from the terminal input
4          ; silo buffer to the current read data buffer.
5          ;
6          ; Inputs:
7          ; R5 = CL unit index number
8          ;
9 002212    IRINGG:
10         ;
11         ; See if this routine is already being used by this unit.
12         ; If so, don't reenter it (the other process will transfer all characters).
13         ;
14 002212    ;          DISABL          ;;;** Disable interrupts **
15 002220    032765 000000G 000000G    BIT      #CM$IRG,CL$STA(R5) ;;;Is this routine already active for unit?
16 002226    001404    BEQ      2$          ;;;Br if not
17 002230    ;          ENABL          ;;;** Enable interrupts **
18 002236    000207    RETURN
19         ;
20         ; This routine is not active, claim it for us
21         ;
22 002240    052765 000000G 000000G 2$:    BIS      #CM$IRG,CL$STA(R5) ;;;Say the routine is now active
23 002246    ;          ENABL          ;;;** Enable interrupts **
24         ;
25         ; Push some registers
26         ;
27 002254    010146    MOV      R1,-(SP)
28 002256    010246    MOV      R2,-(SP)
29 002260    010346    MOV      R3,-(SP)
30 002262    010446    MOV      R4,-(SP)
31         ;
32         ; Get index number of line associated with this CL unit
33         ;
34 002264    016501 000000G    MOV      CL$LIX(R5),R1 ;Get line index number
35         ;
36         ; See if there are any characters in the input buffer and if there
37         ; is a pending read request for this unit.
38         ;
39 002270    3$:    DISABL          ;;;** Disable interrupts **
40 002276    016504 000000G    MOV      CL$RQH(R5),R4 ;;;Is there a pending read request?
41 002302    001475    BEQ      9$          ;;;Br if not
42 002304    032765 000000G 000000G    BIT      #CM$EOF,CL$STA(R5);;;Need to report end of file?
43 002312    001004    BNE      7$          ;;;Br if yes
44 002314    026161 000000G 000000G    CMP      LHIRBS(R1),LHIRBA(R1);;;Any chars in the silo buffer?
45 002322    001465    BEQ      9$          ;;;Br if not
46         ;
47         ; There are characters in the silo buffer and there is a pending
48         ; read request.
49         ;
50 002324    7$:    ENABL          ;;;** Enable interrupts **
51         ;
52         ; See if flag is set which indicates that we should signal end-of-file
53         ;
54 002332    032765 000000G 000000G    BIT      #CM$EOF,CL$STA(R5);Should we signal end of file?
55 002340    001413    BEQ      4$          ;Br if not
56 002342    016403 000000C    MOV      Q.CSW-Q.BLKN(R4),R3;Get pointer to CSW for channel
57 002346    052763 000000G 000000G    BIS      #CS$EOF,C.CSW(R3);Set end of file flag

```

IRINGG -- Move chars from silo buffer to data buffer

```

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        #C0$BNI,CL$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

```

CCIRTN -- Input control character processing routines

```

1          .SBTTL  CCIRTN -- Input control character processing routines
2          ;-----
3          ; These routines are called to process control characters received
4          ; from a line.
5          ;
6          ; Inputs:
7          ;   R2 = Control character
8          ;   R5 = Unit index number
9          ;
10         ; Vector of control character processing routines
11         ;
12         CCIRTN: .WORD  CCINUL          ;00 null
13         .WORD  CCISTR          ;01 SHD
14         .WORD  CCISTR          ;02 STX
15         .WORD  CCISTR          ;03 ETX
16         .WORD  CCISTR          ;04 EDT
17         .WORD  CCISTR          ;05 ENQ
18         .WORD  CCISTR          ;06 ACK
19         .WORD  CCISTR          ;07 BEL
20         .WORD  CCISTR          ;10 BACKSPACE
21         .WORD  CCISTR          ;11 TAB
22         .WORD  CCILF          ;12 LINE FEED
23         .WORD  CCISTR          ;13 VT
24         .WORD  CCISTR          ;14 FF
25         .WORD  CCICR          ;15 CARRIAGE RETURN
26         .WORD  CCISTR          ;16 SO
27         .WORD  CCISTR          ;17 SI
28         .WORD  CCISTR          ;20 DLE
29         .WORD  CCISTR          ;21 XON
30         .WORD  CCISTR          ;22 DC2
31         .WORD  CCISTR          ;23 XOFF
32         .WORD  CCISTR          ;24 DC4
33         .WORD  CCISTR          ;25 NAK
34         .WORD  CCISTR          ;26 SYN
35         .WORD  CCISTR          ;27 ETB
36         .WORD  CCISTR          ;30 CAN
37         .WORD  CCISTR          ;31 EM
38         .WORD  CCICTZ          ;32 SUB (ctrl-Z)

```

CCIRTN -- Input control character processing routines

```

1          ;
2          ; Routine to store the control character
3          ;
4 002612 004737 002706' CCISTR: CALL INPCHR      ;Store the character
5 002616 000207          RETURN
6          ;
7          ; Routine to process a null character
8          ;
9 002620 032765 000000G 000000G CCINUL: BIT      #CO$BNI,CL$OPT(R5);Are we in binary input mode?
10 002626 001371          BNE      CCISTR      ;Br if yes -- go store the null
11 002630 000207          RETURN      ;Discard the null
12         ;
13         ; Routine to process a line feed
14         ;
15 002632 032765 000000G 000000G CCILF:  BIT      #CO$LFI,CL$OPT(R5);Should we ignore input line feeds?
16 002640 001364          BNE      CCISTR      ;Br if not
17 002642 000207          RETURN      ;Discard the LF
18         ;
19         ; Routine to process carriage returns
20         ;
21 002644 016500 000000G          CCICR:  MOV      CL$RQH(R5),R0 ;Get address of current Q element
22 002650 126027 000000C 000000G          CMPB     Q.FUNC-Q.BLKN(R0),#CLSFRL ;Read-line special function?
23 002656 001355          BNE      CCISTR      ;Br if not -- Treat CR as normal char
24 002660 004737 002706'          CALL     INPCHR      ;Store the carriage return
25 002664 004737 002742'          CALL     RDFIN      ;Terminate the read operation
26 002670 000207          RETURN
27         ;
28         ; Routine to process control-Z characters
29         ;
30 002672          CCICTZ:
31         ;
32         ; Set flag which will cause us to return EOF status on next read
33         ;
34 002672 052765 000000G 000000G          BIS      #CM$EOF,CL$STA(R5);Remember EOF has been hit
35         ;
36         ; Terminate this read operation
37         ;
38 002700 004737 002742'          CALL     RDFIN      ;Terminate the read operation
39 002704 000207          RETURN

```

INPCHR -- Move character to user's data buffer

```

1          .SBTTL  INPCHR -- Move character to user's data buffer
2          ;-----
3          ; INPCHR is called to store a data character into the user's buffer
4          ; associated with the current read request.
5          ; If this causes the read request to be completed, the current read
6          ; queue element is returned to the system.
7          ;
8          ; Inputs:
9          ;   R2 = Character to be stored
10         ;   R5 = CL unit index number
11         ;
12 002706 010446 INPCHR: MOV      R4,-(SP)
13         ;
14         ; Get address of current read queue element
15         ;
16 002710 016504 000000G 1$:      MOV      CL$RQH(R5),R4 ;Get pointer to current read queue element
17 002714 001410          BEQ      9$ ;Br if no read request is pending
18         ;
19         ; Store character into data buffer
20         ;
21 002716 010246          MOV      R2,-(SP) ;Stack the data char for PTBYT
22 002720 004737 000000G          CALL     PTBYT ;Move char to user's data buffer
23         ;
24         ; Decrement remaining byte count and see if this completes the read request
25         ;
26 002724 005364 000000C          DEC      Q.WCNT-Q.BLKN(R4);Does this complete the read request?
27 002730 001002          BNE      9$ ;Br if not
28         ;
29         ; The read request is completed.
30         ; Return the queue element to the system.
31         ;
32 002732 004737 002742'          CALL     RDFIN ;Read request is completed
33         ;
34         ; Finished
35         ;
36 002736 012604 9$:      MOV      (SP)+,R4
37 002740 000207          RETURN

```

RDFIN -- Completed a read request

```

1          .SBTTL  RDFIN  -- Completed a read request
2          ;-----
3          ; We have completed a read request.
4          ; Null fill the remainder of the user's buffer if that is needed and then
5          ; call the system I/O completion routine.
6          ;
7          ; Inputs:
8          ;   R5 = CL unit index number.
9          ;
10         RDFIN:  MOV      R3, -(SP)
11                MOV      R4, -(SP)
12          ;
13          ; Get address of current read queue element
14          ;
15         002742  010346  000000G      MOV      CL$RQH(R5), R4      ;Get address of read queue element
16         002744  010446                BEQ      9$                  ;Br if none pending
17          ;
18          ; See if we need to store nulls into the remainder of the buffer
19          ;
20         002754  016504  000000G      MOV      Q.WCNT-Q.BLKN(R4), R3 ;Get remaining byte count
21         002760  001404                BEQ      2$                  ;Br if buffer is full
22         002762  005046      1$:      CLR      -(SP)
23         002764  004737  000000G      CALL     PTBYT                ;Null fill the remainder of the buffer
24         002770  077304                SOB      R3, 1$
25          ;
26          ; Remove the queue element from our internal queue and place on the queue
27          ; of elements waiting to be returned to the system.
28          ;
29         002772                2$:      DISABL                    ;;; ** Disable interrupts **
30         003000  016465  000000C  000000G      MOV      Q.LINK-Q.BLKN(R4), CL$RQH(R5) ;;; Remove Q element from list
31         003006  013764  000004'  000000C      MOV      CQH, Q.LINK-Q.BLKN(R4) ;;; Put Q element on completion list
32         003014  010437  000004'                MOV      R4, CQH
33         003020                ENABL                      ;;; ** Enable interrupts **
34          ;
35          ; Now call system I/O completion routine to free the queue element
36          ;
37         003026  004737  006010'      CALL     RTNQ                ;Return queue element to the system
38          ;
39          ; Finished
40          ;
41         003032  012604      9$:      MOV      (SP)+, R4
42         003034  012603                MOV      (SP)+, R3
43         003036  000207                RETURN

```

CLTIMR -- Routine called from clock interrupt routine

```

1          .SBTTL  CLTIMR -- Routine called from clock interrupt routine
2          ;-----
3          ; CLTIMR is called on a clock interrupt (50/60 Hz) basis to move characters
4          ; to/from the user's I/O data buffer and the output/input CL character
5          ; ring buffers. We do this type of processing on a clock interrupt
6          ; basis to avoid having to do a .FORK on each input/output character
7          ; interrupt.
8          ;
9 003040    010146    CLTIMR: MOV      R1,-(SP)
10 003042    010446    MOV      R4,-(SP)
11 003044    010546    MOV      R5,-(SP)
12          ;
13          ; Begin loop to service each CL unit
14          ;
15 003046    012705    000000C    MOV      #2*(CLTOTL-1),R5;Get index # of last CL unit
16          ;
17          ; See if this CL unit is connected to a line
18          ;
19 003052    016501    000000G    1$: MOV      CL$LIX(R5),R1    ;Is this CL unit connected to a line?
20 003056    001412    BEQ      2$          ;Br if not
21          ;
22          ; See if user wants to change status of Data Terminal Ready
23          ;
24 003060    004737    006250'    CALL      SETDTR          ;Call routine to set or clear the DTR flag
25          ;
26          ; Call DRINGP for each line to try to move characters from the user's buffer
27          ; to the output ring buffer.
28          ;
29 003064    005765    000000G    TST      CL$XLN(R5)        ;Is this CL unit cross connected to TT line?
30 003070    001403    BEQ      3$          ;Br if not
31 003072    004737    004574'    CALL      CLOCPY          ;Copy characters to CL output ring buffer
32 003076    000402    BR      2$
33 003100    004737    003130'    3$: CALL      DRINGP          ;Move chars to output ring buffer
34          ;
35          ; Process the next CL unit
36          ;
37 003104    ENABL          ;Make sure interrupts are enabled
38 003112    162705    0000002    SUB      #2,R5          ;Get index of next line
39 003116    002355    BGE      1$          ;Loop if more lines to service
40          ;
41          ; Finished
42          ;
43 003120    012605    MOV      (SP)+,R5
44 003122    012604    MOV      (SP)+,R4
45 003124    012601    MOV      (SP)+,R1
46 003126    000207    RETURN

```

ORINGP -- Move chars from data buffer to output ring buffer

```

1          .SBTTL  ORINGP -- Move chars from data buffer to output ring buffer
2          ;-----
3          ; ORINGP is called to move characters from the current output data buffer
4          ; to the output ring buffer.
5          ;
6          ; Inputs:
7          ; R5 = CL unit index number
8          ;
9 003130 010246 ORINGP: MOV      R2,-(SP)
10 003132 010346      MOV      R3,-(SP)
11          ;
12          ; See if this routine is already being used by this unit.
13          ; If so, don't reenter it (the other process will transfer all characters
14          ; that can be transferred).
15          ;
16 003134          DISABL          ;** Disable interrupts **
17 003142 032765 000000G 000000G BIT      #CM$ORP,CL$STA(R5);;Is this routine already active for unit?
18 003150 001402      BEQ      21$          ;Br if not
19 003152 000137 003570'      JMP      9$          ;Br if routine already active
20          ;
21          ; This routine is not active for this unit. Claim it.
22          ;
23 003156 052765 000000G 000000G 21$: BIS      #CM$ORP,CL$STA(R5);;Say routine is now active
24 003164          ENABL          ;** Enable interrupts **
25 003172 005002      11$: CLR      R2          ;Count # chars moved to output ring buffer
26          ;
27          ; See if there is any free space in the output ring buffer and see if
28          ; there is a pending write request for this unit.
29          ;
30 003174          4$: DISABL          ;** Disable interrupts **
31 003202 005765 000000G      TST      CL$ORS(R5)          ;Any available space in ring buffer?
32 003206 001555      BEQ      8$          ;Br if no space available
33 003210 005765 000000G      TST      CL$WQH(R5)          ;Is there a pending write request?
34 003214 001004      BNE      20$          ;Br if a write is pending
35 003216 032765 000000G 000000G      BIT      #CM$EFP,CL$STA(R5);;Are we doing end-of-file processing?
36 003224 001546      BEQ      8$          ;Br if not
37          ;
38          ; There is free space in the output ring buffer and there is a pending
39          ; write request.
40          ; We will move characters from the user's buffer to the output ring buffer.
41          ;
42 003226          20$: ENABL          ;** Enable interrupts **
43          ;
44          ; See if we are sending spaces to simulate tabs
45          ;
46 003234 032765 000000G 000000G 15$: BIT      #CM$TBS,CL$STA(R5);Are we doing tab simulation?
47 003242 001412      BEQ      16$          ;Br if not
48 003244 032765 000007 000000G      BIT      #7,CL$COL(R5) ;Have we reached the next tab stop?
49 003252 001403      BEQ      2$          ;Br if yes
50 003254 012700 000040      MOV      #SPACE,R0          ;Get space for simulation
51 003260 000474      BR      12$
52 003262 042765 000000G 000000G 2$: BIC      #CM$TBS,CL$STA(R5);Say we are finished with tab simulation
53          ;
54          ; See if we are sending line feeds to simulate a form feed
55          ;
56 003270 032765 000000G 000000G 16$: BIT      #CM$FFS,CL$STA(R5);Are we doing form feed simulation?
57 003276 001414      BEQ      1$          ;Br if not

```

ORINGP -- Move chars from data buffer to output ring buffer

```

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' 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$:   BIT      #CO$BND,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$BBT,CL$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

```

ORINGP -- Move chars from data buffer to output ring buffer

```

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          MOV      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

```

GETCHR -- Get next output char from user's data buffer

```

1          .SBTTL  GETCHR -- Get next output char from user's data buffer
2          -----
3          ; GETCHR is called to obtain the next character from the user's
4          ; data buffer.
5          ;
6          ; Inputs:
7          ;   R5 = CL unit index number
8          ;
9          ; Outputs:
10         ;   C-flag cleared ==> A character was gotten
11         ;   C-flag set      ==> No more characters are available
12         ;   R0 = Character gotten if C-flag is cleared
13         ;
14 003604 010446 GETCHR: MOV      R4, -(SP)
15         ;
16         ; See if we should do end-of-file output processing.
17         ;
18 003606 032765 000000G 000000G 5$:      BIT      #CM$EFP, CL$STA(R5) ; Should we do end-of-file processing?
19 003614 001403      BEQ      2$          ; Br if not
20         ;
21         ; We are doing end-of-file output processing.
22         ; See if there is another end-of-file character to send.
23         ;
24 003616 004737 003760'      CALL     EOFCHR          ; See if another eof char to send
25 003622 103054      BCC      12$          ; Br if we got an EOF character
26         ;
27         ; See if there is a pending write operation
28         ;
29 003624 016504 000000G 2$:      MOV      CL$WQH(R5), R4 ; Get pointer to current write queue element
30 003630 001446      BEQ      10$          ; Br if no pending write operation
31         ;
32         ; If the FORMO option is in effect and this is the first write to
33         ; block 0, send a form feed.
34         ;
35 003632 005764 000000C      TST      Q.BLKN-Q.BLKN(R4); Is block number = 0?
36 003636 001011      BNE      4$          ; Br if not
37 003640 032765 000000G 000000G      BIT      #CO$FFO, CL$OPT(R5) ; Is the FORMO option in effect?
38 003646 001405      BEQ      4$          ; Br if not
39 003650 005264 000000C      INC      Q.BLKN-Q.BLKN(R4); Inc block # so we only do this once
40 003654 112700 000014      MOVB     #FF, R0      ; Get form feed character
41 003660 000434      BR        9$          ; Return the form feed
42         ;
43         ; See if current queue element has another character to be sent
44         ;
45 003662 005764 000000C 4$:      TST      Q.WCNT-Q.BLKN(R4); Any remaining bytes to send?
46 003666 001406      BEQ      3$          ; Br if not -- write request is finished
47         ;
48         ; Get next character from user's buffer
49         ;
50 003670 005364 000000C      DEC      Q.WCNT-Q.BLKN(R4); Decrease remaining byte count
51 003674 004737 000000G      CALL     GTBYT          ; Get next byte from user's buffer
52 003700 012600      MOV      (SP)+, R0      ; Get the returned character
53 003702 000423      BR        9$          ; Return the character
54         ;
55         ; This write operation is completed.
56         ; Remove the queue element from our internal queue and place it
57         ; on the queue of elements waiting to be returned to the system.

```

GETCHR -- Get next output char from user's data buffer

```
58 ;
59 003704 3$: DISABL ;;;** Disable interrupts **
60 003712 016465 000000C 000000G MOV Q.LINK-Q.BLKN(R4),CL$WQH(R5) ;;;Remove element from internal Q
61 003720 013764 000004' 000000C MOV CQH,Q.LINK-Q.BLKN(R4) ;;;Add to list of completed requests
62 003726 010437 000004' MOV R4,CQH
63 003732 ENABL ;;;** Enable interrupts **
64 ;
65 ; Return the completed queue element to the system (do .DRFIN)
66 ;
67 003740 004737 006010' CALL RTNQ ;Tell system we finished the operation
68 ;
69 ; Go back and see if there is another write request pending
70 ;
71 003744 000727 BR 2$ ;Go check for another write request
72 ;
73 ; There are no available characters
74 ;
75 003746 000261 10$: SEC ;Signal that no chars are available
76 003750 000401 BR 12$
77 ;
78 ; We got a character
79 ;
80 003752 000241 9$: CLC ;Signal that we got a character
81 ;
82 ; Finished
83 ;
84 003754 012604 12$: MOV (SP)+,R4
85 003756 000207 RETURN
```

EOFCHR -- Get next end-of-file output character

```

1          .SBTTL  EOFCHR -- Get next end-of-file output character
2          ;-----
3          ; This routine is called during end-of-file output processing to see
4          ; if there is another end-of-file output character to send.
5          ;
6          ; Inputs:
7          ;   R5 = CL unit index number
8          ;
9          ; Outputs:
10         ;   C-flag cleared ==> Got a character
11         ;   C-flag set ==> No more characters
12         ;   R0 = Character gotten if C-flag cleared.
13         ;
14 003760   EOFCHR:
15         ;
16         ; See if we need to send form-feeds
17         ;
18 003760   126565 000001G 000000G      CMPB    CL$EPN+1(R5),CL$EPN(R5);Do we need to send more form-feeds?
19 003766   103005                      BHIS    1$                ;Br if not
20 003770   105265 000001G              INCB    CL$EPN+1(R5)        ;Count another form-feed being sent
21 003774   012700 000014              MOV     #FF,R0              ;Get form-feed character
22 004000   000422                      BR      7$                ;Go send it
23         ;
24         ; See if we need to send characters from ENDSTRING
25         ;
26 004002   016500 000000G      1$:      MOV     CL$EPP(R5),R0      ;Are we sending end-string characters?
27 004006   001405                      BEQ     2$                ;Br if not
28 004010   111000                      MOVB    (R0),R0            ;Get next char to send
29 004012   001403                      BEQ     2$                ;Br if reached end of string
30 004014   005265 000000G      INC     CL$EPP(R5)                ;Advance character pointer
31 004020   000412                      BR      7$                ;Go send the character
32         ;
33         ; We have finished all end-of-file output processing
34         ;
35 004022   105065 000001G      2$:      CLRB    CL$EPN+1(R5)      ;Reset form-feed count
36 004026   016565 000000G 000000G      MOV     CL$EPS(R5),CL$EPP(R5);Reset end-string pointer
37 004034   042765 000000G 000000G      BIC     #CM$EFP,CL$STA(R5);Finished end-of-file output processing
38 004042   000261                      SEC                      ;Signal that no character was gotten
39 004044   000401                      BR      9$
40         ;
41         ; We got a character
42         ;
43 004046   000241      7$:      CLC                      ;Signal that we got a character
44         ;
45         ; Finished
46         ;
47 004050   000207      9$:      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, RO 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         CCORTN: .WORD  CCONUL          ;00 null
11                .WORD  CCOCTL          ;01 SH0
12                .WORD  CCOCTL          ;02 STX
13                .WORD  CCOCTL          ;03 ETX
14                .WORD  CCOCTL          ;04 EOT
15                .WORD  CCOCTL          ;05 ENQ
16                .WORD  CCOCTL          ;06 ACK
17                .WORD  CCOCTL          ;07 BEL
18                .WORD  CCOBS           ;10 BACKSPACE
19                .WORD  CCOTAB          ;11 TAB
20                .WORD  CCOLF           ;12 LINE FEED
21                .WORD  CCOCTL          ;13 VT
22                .WORD  CCOFF          ;14 FF
23                .WORD  CCOCR          ;15 CARRIAGE RETURN
24                .WORD  CCOCTL          ;16 SO
25                .WORD  CCOCTL          ;17 SI
26                .WORD  CCOCTL          ;20 DLE
27                .WORD  CCOCTL          ;21 DC1 (ctrl-Q)
28                .WORD  CCOCTL          ;22 DC2
29                .WORD  CCOCTL          ;23 DC3 (ctrl-S)
30                .WORD  CCOCTL          ;24 DC4
31                .WORD  CCOCTL          ;25 NAK
32                .WORD  CCOCTL          ;26 SYN
33                .WORD  CCOCTL          ;27 ETB
34                .WORD  CCOCTL          ;30 CAN
35                .WORD  CCOCTL          ;31 EM
36                .WORD  CCOCTL          ;32 SUB (ctrl-Z)
37                .WORD  CCOCTL          ;33 ESC
38                .WORD  CCOCTL          ;34 FS
39                .WORD  CCOCTL          ;35 GS
40                .WORD  CCOCTL          ;36 RS
41                .WORD  CCOCTL          ;37 US

```

CCORTN -- Output control character processing routines

```

1          ;
2          ; Process a general control character
3          ;
4 004152 042765 000000G 000000G CCCTL: BIC      #CM$CRL,CL$STA(R5) ; Say last char out was not carriage return
5 004160 032765 000000G 000000G      BIT      #CD$CTL,CL$OPT(R5) ; Are we to transmit control chars?
6 004166 001002      BNE      CCOSND      ; Br if yes
7 004170 000261      SEC                  ; Say to ignore this character
8 004172 000207      RETURN
9          ;
10         ; Routine to cause the current control character to be transmitted unchanged
11         ;
12 004174 000241 CCOSND: CLC                  ; Say to send the character
13 004176 000207      RETURN
14         ;
15         ; Process null character
16         ;
17 004200 000261 CCONUL: SEC                  ; Say to ignore this character
18 004202 000207      RETURN
19         ;
20         ; Process Backspace character
21         ;
22 004204 042765 000000G 000000G CCBS:  BIC      #CM$CRL,CL$STA(R5) ; Say last char out was not carriage return
23 004212 005365 000000G      DEC      CL$COL(R5) ; Say we are moving back 1 char
24 004216 002366      BGE      CCOSND      ; Br if did not go past column 0
25 004220 005065 000000G      CLR      CL$COL(R5) ; Constrain to column 0
26 004224 000763      BR       CCOSND      ; Go send the character
27         ;
28         ; Process tab character
29         ;
30 004226 042765 000000G 000000G CCOTAB: BIC      #CM$CRL,CL$STA(R5) ; Say last char out was not carriage return
31 004234 032765 000000G 000000G      BIT      #CD$TAB,CL$OPT(R5) ; Does device have hardware tab support
32 004242 001416      BEQ      1$          ; Br if not
33 004244 062765 000010 000000G      ADD      #8,CL$COL(R5) ; Bound up to next tab stop
34 004252 042765 000007 000000G      BIC      #7,CL$COL(R5)
35 004260 005765 000000G      TST      CL$WID(R5) ; Was a maximum width specified?
36 004264 001743      BEQ      CCOSND      ; Br if not -- go send the tab
37 004266 026565 000000G 000000G      CMP      CL$COL(R5),CL$WID(R5) ; Have we gone beyond max width?
38 004274 103737      BLO      CCOSND      ; Br if not
39 004276 000740      BR       CCONUL      ; Discard this tab
40 004300 052765 000000G 000000G 1$:  BIS      #CM$TBS,CL$STA(R5) ; Say we are doing tab simulation
41 004306 005265 000000G      INC      CL$COL(R5) ; Advance column counter
42 004312 012700 000040      MOV      #SPACE,R0 ; Send a space character
43 004316 000726      BR       CCOSND
44         ;
45         ; Process Line feed character
46         ;
47 004320 005265 000000G      COLF:  INC      CL$LIN(R5) ; Increment line-on-page counter
48 004324 016500 000000G      MOV      CL$LEN(R5),R0 ; Was a page length value specified?
49 004330 001431      BEQ      5$          ; Br if not
50 004332 026500 000000G      CMP      CL$LIN(R5),R0 ; Have we reached the top of a new page?
51 004336 103405      BLO      2$          ; Br if not
52 004340 005065 000000G      CLR      CL$LIN(R5) ; Say we are at top of a new page
53 004344 042765 000000G 000000G      BIC      #CM$FFS,CL$STA(R5) ; Stop doing form feed simulation
54 004352 166500 000000G      2$:  SUB      CL$SKP(R5),R0 ; See if we are to skip lines at bottom of page
55 004356 026500 000000G      CMP      CL$LIN(R5),R0 ; Have we reached the skip point?
56 004362 001014      BNE      5$          ; Br if not
57 004364 032765 000000G 000000G      BIT      #CM$FFS,CL$STA(R5) ; Are we already doing form feed simulation?

```

```

58 004372 001010          BNE      5$          ;Br if yes
59 004374 112700 000014    MOVB     #FF,RO      ;At skip point -- Do a form feed
60 004400 052765 000000G 000000G    BIS      #CM$FFI,CL$STA(R5) ;Ignore FF if 1st char after skip
61 004406 005365 000000G    DEC      CL$LIN(R5) ;Set line counter back -- haven't sent LF yet
62 004412 000420          BR       CCOFF        ;Go process the form feed
63 004414 032765 000000G 000000G 5$:  BIT      #CO$LFO,CL$OPT(R5) ;Should we discard line feeds on output?
64 004422 001010          BNE      6$          ;Br if not
65 004424 032765 000000G 000000G    BIT      #CM$CRL,CL$STA(R5);Was last char out a carriage return?
66 004432 001404          BEQ      6$          ;Br if not
67 004434 042765 000000G 000000G    BIC      #CM$CRL,CL$STA(R5);Clear flag that says carriage return last
68 004442 000656          BR       CCONUL       ;Discard the line feed
69 004444 112700 000012      6$:    MOVB     #LF,RO      ;Get back line feed character
70 004450 000241          CLC              ;Say to send it
71 004452 000207      9$:    RETURN
72                                ;
73                                ; Process Form feed character
74                                ;
75 004454 042765 000000G 000000G CCOFF: BIC      #CM$CRL,CL$STA(R5) ;Say last char out was not carriage return
76 004462 032765 000000G 000000G    BIT      #CO$FF,CL$OPT(R5) ;Does this device support form feed chars?
77 004470 001403          BEQ      1$          ;Br if not
78 004472 005065 000000G    CLR      CL$LIN(R5) ;Say we are at top of the page
79 004476 000636          BR       CCOSND       ;Go send the form feed
80 004500 005765 000000G      1$:    TST      CL$LEN(R5) ;Do we have a non-zero page length?
81 004504 001406          BEQ      2$          ;If not then discard the FF
82 004506 052765 000000G 000000G    BIS      #CM$FFS,CL$STA(R5);Say we are starting form-feed simulation
83 004514 012700 000012      MOV      #LF,RO      ;Translate form feed to line feed
84 004520 000677          BR       CCOLF        ;Go send line feed
85 004522 005065 000000G      2$:    CLR      CL$LIN(R5) ;Say we are at top of page
86 004526 000624          BR       CCONUL       ;Discard the character
87                                ;
88                                ; Process carriage return character
89                                ;
90 004530 052765 000000G 000000G CCOCR: BIS      #CM$CRL,CL$STA(R5) ;Say last char out was carriage return
91 004536 005065 000000G    CLR      CL$COL(R5) ;Say we are back to column 0
92 004542 032765 000000G 000000G    BIT      #CO$CR,CL$OPT(R5);Should we transmit carriage returns?
93 004550 001211          BNE      CCOSND       ;Br if yes
94 004552 000261          SEC              ;Ignore this char
95 004554 000207          RETURN

```

CLXICP -- Got char for output to cross connected CL line

```

1          .SBTTL  CLXICP -- Got char for output to cross connected CL line
2          -----
3          ; CLXICP is called at fork level when a character is received from a
4          ; TT line that is cross connected to a CL line.
5          ; It copies all possible characters from the input silo of the TT line
6          ; to the output silo for the CL line and initiates output to the CL line.
7          ;
8          ; Inputs:
9          ;   R1 = Index number of TT line that received the character
10         ;
11 004556 010546 CLXICP: MOV      R5, -(SP)
12         ;
13         ;   Get CL index of line we are cross connected to
14         ;
15 004560 016105 000000G      MOV      LXCL(R1), R5      ; Get # of CL line we are connected to
16         ;
17         ;   Call routine to copy all chars from TT input silo to CL output silo
18         ;
19 004564 004737 004574'      CALL      CLOCPY          ; Copy chars to CL output silo
20         ;
21         ;   Finished
22         ;
23 004570 012605      MOV      (SP)+, R5
24 004572 000207      RETURN

```

CLOCPY -- Copy characters from TT input buf to CL output buf

```

1          .SBTTL  CLOCPY -- Copy characters from TT input buf to CL output buf
2          -----
3          ; CLOCPY is called to copy characters from the input silo of a TT line to
4          ; the output buffer of a cross-connected CL line.
5          ;
6          ; Inputs:
7          ;   R5 = Unit index of CL line
8          ;
9          CLOCPY: MOV     R1, -(SP)
10         MOV     R2, -(SP)
11         MOV     R3, -(SP)
12         ;
13         ; See if this routine is already being used by this unit.
14         ; If so, don't reenter it (the other process will transfer all
15         ; characters that can be transferred).
16         ;
17         004602         DISABL                    ;;; Disable interrupts
18         004610 032765 000000G 000000G         BIT      #CM$ORP, CL$STA(R5) ;;; Is this routine already active?
19         004616 001113         BNE      9$          ;;; Br if yes
20         ;
21         ; This routine is not active for this unit. Claim it.
22         ;
23         004620 052765 000000G 000000G         BIS      #CM$ORP, CL$STA(R5) ;;; Say routine is now active
24         004626         ENABL                    ; Enable interrupts
25         004634 005002 11$: CLR      R2          ; Count # chars copied to output buffer
26         ;
27         ; See if cross-connection is still in effect
28         ;
29         004636 016501 000000G 4$: MOV     CL$XLN(R5), R1 ; Get number of cross-connected TT line
30         004642 001476         BEQ      10$         ; Br if no longer cross connected
31         ;
32         ; See if there is any free space in the output ring buffer.
33         ;
34         004644         DISABL                    ;;; Disable interrupts
35         004652 005765 000000G         TST      CL$ORS(R5) ;;; Any available space in ring buffer?
36         004656 001460         BEQ      8$          ;;; Br if no space available
37         004660 026161 000000G 000000G         CMP      LHIRBS(R1), LHIRBA(R1) ;;; Any chars in TT input silo?
38         004666 001454         BEQ      8$          ;;; Br if not
39         004670         ENABL                    ; Enable interrupts
40         ;
41         ; Get next character from TT input silo
42         ;
43         004676 004777 000000G         CALL     @SILFET ; Get next char from TT input silo
44         004702 103446         BCS      8$          ; Br if no more chars available
45         ;
46         ; We got a character.
47         ; See if character has special significance.
48         ;
49         004704 032765 000000G 000000G         BIT      #CM$MCC, CL$STA(R5) ; Modem control or literal char?
50         004712 001407         BEQ      1$          ; Br if not
51         004714 042765 000000G 000000G         BIC      #CM$MCC, CL$STA(R5) ; Reset literal-character flag
52         004722 004737 005064'         CALL     CLXMCC ; Process the character
53         004726 103743         BCS      4$          ; Br if finished with char
54         004730 000415         BR       2$          ; Go transmit the character
55         004732 120037 000000G 1$: CMPB     R0, VCXTRM ; Control-\ -- Terminate connection?
56         004736 001003         BNE      3$          ; Br if not
57         004740 004737 005412'         CALL     CLXBRK ; Break cross connection and drop DTR

```

CLDCPY -- Copy characters from TT input buf to CL output buf

```

58 004744 000425          BR      8$          ;Finished
59 004746 120037 000000G  3$:  CMPB    RO,V0XCTL      ;Control-A means next char is modem control
60 004752 001004          BNE     2$          ;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    RO,(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

```

```

1          .SBTTL  CLXMCC -- Process cross connect modem control character
2          ;-----
3          ; Process a modem control character for a cross connection.
4          ;
5          ; Inputs:
6          ;   R0 = Character
7          ;   R5 = CL unit index number
8          ;
9          ; Outputs:
10         ;   C-flag cleared ==> Go ahead and transmit this character.
11         ;   C-flag set    ==> Do not transmit this character.
12         ;
13 005064 010046 CLXMCC: MOV     R0,-(SP)      ; Save the character
14 005066 010146      MOV     R1,-(SP)
15         ;
16         ; Translate lower-case to upper-case
17         ;
18 005070 120027 000141      CMPB    R0,#141      ; Is this a lower-case letter?
19 005074 103405      BLO     1$                ; Br if not
20 005076 120027 000172      CMPB    R0,#172      ;
21 005102 101002      BHI     1$                ; Br if not
22 005104 162700 000040      SUB     #40,R0       ; Convert to upper-case
23         ;
24         ; "B" -- Start sending a break
25         ;
26 005110 120027 000102 1$:  CMPB    R0,#'B      ; Is character B?
27 005114 001042      BNE     2$                ; Br if not
28 005116 052765 000000G 000000G  BIS     #CM$BRK,CL$STA(R5); Set flag saying we are sending break
29 005124 012700 000000G      MOV     #MS$BRK,R0  ; Set flag to start break transmission
30 005130 004737 006346'      CALL    SETBRK     ; Call hardware routine to start sending break
31 005134 004737 005514'      CALL    CLSTRT     ; Start transmitter
32 005140 004737 000000G      CALL    GETRTQ     ; Get a real-time queue element (ptr in R1)
33 005144 012761 000036 000000G  MOV     #30,CQ$LOD(R1); Set approx 0.5 second time interval
34 005152 012761 005362' 000000G  MOV     #CLXSSB,CQ$RTN(R1); Set address of compl routine
35 005160 013761 000000G 000000G  MOV     @#KPAR5,CQ$PA5(R1); Save system par 5 mapping
36 005166 010561 000000G      MOV     R5,CQ$RO(R1) ; Set CL unit index
37 005172      DISABL                     ;;; * Disable interrupts *
38 005200 013761 000000G 000000G  MOV     MRKTHD,CQ$LNK(R1);;; Put new element on linked list
39 005206 010137 000000G      MOV     R1,MRKTHD  ;;;
40 005212      ENABL                      ; * Enable interrupts *
41 005220 000452      BR      20$
42         ;
43         ; "D" -- Raise DTR
44         ;
45 005222 120027 000104 2$:  CMPB    R0,#'D      ; Is character D?
46 005226 001006      BNE     4$                ; Br if not
47 005230 052765 000000G 000000G  BIS     #CD$DTR,CL$OPT(R5); Request DTR up
48 005236 004737 006250'      CALL    SETDTR     ; Call routine to raise DTR
49 005242 000441      BR      20$
50         ;
51         ; "H" -- Drop DTR
52         ;
53 005244 120027 000110 4$:  CMPB    R0,#'H      ; Is character H?
54 005250 001006      BNE     5$                ; Br if not
55 005252 042765 000000G 000000G  BIC     #CD$DTR,CL$OPT(R5); Request DTR drop
56 005260 004737 006250'      CALL    SETDTR     ; Call routine to drop DTR
57 005264 000430      BR      20$

```

```

58      ;
59      ; "R" -- Reset XON/XOFF status
60      ;
61 005266 120027 000122      5$:      CMPB      RO,#'R      ;Reset XON/XOFF status?
62 005272 001017              BNE          6$          ;Br if not
63 005274 016501 000000G      MOV          CL$LIX(R5),R1    ;Get index of line we are connected to
64 005300 042761 000000G 000000G BIC          #$CTRLS,LSW3(R1);Reset XOFF received flag
65 005306 042761 000000G 000000G BIC          #$HISTP,LSW10(R1);Say input has not been stopped by XOFF
66 005314 016100 000000G      MOV          LCDTYP(R1),RO    ;Get device type code
67 005320 004770 000000G      CALL         @CDSXON(RO)      ;Call routine to stuff XON into output
68 005324 004737 005514'      CALL         CLSTRT          ;Try to start output to CL unit
69 005330 000406              BR           20$
70      ;
71      ; "X" -- Break cross connection without dropping DTR
72      ;
73 005332 120027 000130      6$:      CMPB      RO,#'X      ;Break cross-connection?
74 005336 001005              BNE          21$          ;Br if not
75 005340 004737 005432'      CALL         CLXDRP          ;Break cross connection
76 005344 000400              BR           20$          ;Finished with character
77      ;
78      ; This is a modem control character.
79      ; Don't send it.
80      ;
81 005346 000261      20$:      SEC                      ;Signal not to send the character
82 005350 000401              BR           22$
83      ;
84      ; This is not a modem control character.
85      ; Send the character literally.
86      ;
87 005352 000241      21$:      CLC                      ;Signal that we should send the character
88      ;
89      ; Finished
90      ;
91 005354 012601      22$:      MOV          (SP)+,R1
92 005356 012600              MOV          (SP)+,RO
93 005360 000207              RETURN

```

CLXMCC -- Process cross connect modem control character

```
1 ;-----  
2 ; System completion routine called to stop sending break to a  
3 ; cross-connected CL line.  
4 ;  
5 ; Inputs:  
6 ; RO = CL unit index  
7 ;  
8 005362 010546 CLXSSB: MOV R5, -(SP)  
9 005364 010005 MOV RO, R5 ;Get CL unit index  
10 ;  
11 ; Stop sending break  
12 ;  
13 005366 005000 CLR RO ;Clear break-send flag  
14 005370 004737 006346' CALL SETBRK ;Call hardware routine to end break  
15 005374 042765 000000G 000000G BIC #CM$BRK, CL$STA(R5); Clear break-sending flag  
16 005402 004737 005514' CALL CLSTRT ;Start transmitter  
17 ;  
18 ; Finished  
19 ;  
20 005406 012605 MOV (SP)+, R5  
21 005410 000207 RETURN
```

CLXBRK -- Break a CL-TT cross connection and drop DTR

```
1          .SBTTL  CLXBRK -- Break a CL-TT cross connection and drop DTR
2          ;-----
3          ; This routine is called when we receive control-\ to break the cross
4          ; connection between a CL unit and a TT line.
5          ; In addition to breaking the connection, DTR is dropped to hang up.
6          ;
7          ; Inputs:
8          ;   R5 = CL unit index
9          ;
10         005412 CLXBRK:
11         ;
12         ; First, drop DTR
13         ;
14         005412 042765 000000G 000000G      BIC      #CD$DTR,CL$OPT(R5) ;Request DTR drop
15         005420 004737 006250'      CALL      SETDTR      ;Call routine to drop DTR
16         ;
17         ; Now break the cross connection
18         ;
19         005424 004737 005432'      CALL      CLXDRP      ;Break the cross connection
20         ;
21         ; Finished
22         ;
23         005430 000207      RETURN
```

CLXDRP -- Break a CL-TT cross connection

```

1          .SBTTL  CLXDRP -- Break a CL-TT cross connection
2          ;-----
3          ; This routine is called when we receive control-\ to break the cross
4          ; connection between a CL unit and a TT line.
5          ; DTR is not dropped by this routine. Call CLXBRK to drop DTR too.
6          ;
7          ; Inputs:
8          ;   R5 = CL unit index
9          ;
10         005432  010146  CLXDRP: MOV      R1,-(SP)
11         ;
12         ; Reset this CL unit
13         ;
14         005434  004737  001764'  CALL      CLREST      ;Reset the CL unit
15         ;
16         ; Reconnect time-sharing line to normal input character processing routine
17         ;
18         005440          DISABL          ;;; ** Disable interrupts **
19         005446  016501  000000G  MOV      CL$XLN(R5),R1  ;;; Get number of cross-connected TT line
20         005452  012761  177777  000000G  MOV      #-1,LXCL(R1)  ;;; Say not connected to a CL unit
21         005460  012761  000000G  000000G  MOV      #TTINCP,LINIR(R1);;; Connect to TT input processing routine
22         ;
23         ; Say CL unit no longer connected to time-sharing line
24         ;
25         005466  005065  000000G  CLR      CL$XLN(R5)      ;;; CL unit no longer connected to TT line
26         005472          ENABL          ;;; ** Enable interrupts **
27         ;
28         ; Restart the execution of the job
29         ;
30         005500  116101  000000G  MOVB     LNMAP(R1),R1     ;Get virtual job index number
31         005504  004737  000000G  CALL     FORCEX          ;Cause job to continue execution
32         ;
33         ; Finished
34         ;
35         005510  012601          MOV      (SP)+,R1
36         005512  000207          RETURN

```

CLSTRT -- Start transmissions to a line

```

1          .SBTTL  CLSTRT -- Start transmissions to a line
2          ;-----
3          ; CLSTRT is called to initiate transmission to a line.
4          ;
5          ; Inputs:
6          ;   R5 = CL unit index number.
7          ;
8 005514 010146 CLSTRT: MOV      R1,-(SP)
9          ;
10         ; Convert CL unit number into line index number
11         ;
12 005516 016501 000000G      MOV      CL$LIX(R5),R1  ;Get line index # for this CL unit
13         ;
14         ; Call device dependent routine to start the transmitter
15         ;
16 005522 016100 000000G      MOV      LCDTYP(R1),R0  ;Get communications device type code
17 005526 004770 000000G      CALL     @CDSTRT(R0)    ;Call device dependent startup routine
18 005532 005237 000000G      INC      NEDCDO        ;Say output character processing needed
19 005536 005237 000000G      INC      NEDCLO        ;Say CL output processing needed
20         ;
21         ; Finished
22         ;
23 005542 012601      MOV      (SP)+,R1
24 005544 000207      RETURN

```

CLABRT -- Handler abort routine

```

1          .SBTTL  CLABRT -- Handler abort routine
2          ;-----
3          ; CLABRT is jumped to from the handler abort entry point.
4          ; It terminates any I/O operations for the job being aborted.
5          ;
6          ; Inputs:
7          ;   R4 = Aborted job index number / 2
8          ;
9          CLABRT: MOV     R3, -(SP)
10         MOV     R4, -(SP)
11         MOV     R5, -(SP)
12         ;
13         ; Check each CL unit to see if there are any requests for this job
14         ;
15         MOV     #2*(CLTOTL-1), R5; Get index to last CL unit
16         1$: MOV     #CL$RQH, R3    ; Get address of read queue head
17         CALL    CKABTQ             ; See if there are any entries on this queue
18         MOV     #CL$WQH, R3    ; Get address of write queue head
19         CALL    CKABTQ             ; See if there are any entries on this queue
20         SUB     #2, R5            ; Get index number of next CL unit
21         BGE     1$                ; Br if more units to check
22         ;
23         ; Call routine to return any freed queue elements to the system
24         ;
25         CALL    RTNQ              ; Return freed queue elements to the system
26         ;
27         ; Finished
28         ;
29         MOV     (SP)+, R5
30         MOV     (SP)+, R4
31         MOV     (SP)+, R3
32         RETURN

```

CKABTQ -- Check for aborted queue elements

```

1          SBTTL  CKABTQ -- Check for aborted queue elements
2          -----
3          ; CKABTQ is called to check to see if any queue elements belonging to
4          ; an aborted job are on a specified internal queue.
5          ; If any queue elements for the aborted job are found, they are placed
6          ; on the completion queue list.
7          ;
8          ; Inputs:
9          ;   R3 = Pointer to base of queue head vector for CL units.
10         ;   R4 = # of job being aborted
11         ;   R5 = CL unit index number of queue to check.
12         ;
13 005622 010246 CKABTQ: MOV      R2,-(SP)
14 005624 010346      MOV      R3,-(SP)
15 005626 010546      MOV      R5,-(SP)
16         ;
17         ; Get address of queue head
18         ;
19 005630 060305      ADD      R3,R5          ;Point to queue head for this unit
20 005632 162705 000000C SUB      #Q.LINK-Q.BLKN,R5;Make head look like fake queue entry
21         ;
22         ; Search for entries in the queue
23         ;
24 005636 010503 1$:    MOV      R5,R3          ;Point to queue head
25 005640      DISABL                      ;** Disable interrupts **
26 005646 010302      MOV      R3,R2          ;;; Save address of current entry
27 005650 016303 000000C 2$:    MOV      Q.LINK-Q.BLKN(R3),R3; ;Get address of next entry
28 005654 001417      BEQ      9$              ;;; Br if no entries for job being aborted
29 005656 120463 000000C      CMPB   R4,Q.JOB-Q.BLKN(R3); ;Is this the job being aborted?
30 005662 001372      BNE      2$              ;;; Keep looking if not
31         ;
32         ; We found an entry for the job being aborted
33         ; Remove it from our internal queue and place on the completion queue
34         ;
35 005664 016362 000000C 000000C      MOV      Q.LINK-Q.BLKN(R3),Q.LINK-Q.BLKN(R2); ;Remove from list
36 005672 013763 000004' 000000C      MOV      CQH,Q.LINK-Q.BLKN(R3); ;Put on completion list
37 005700 010337 000004'      MOV      R3,CQH
38         ;
39         ; Go back and see if there are any more entries to remove
40         ;
41 005704      ENABL                      ;** Enable interrupts **
42 005712 000751      BR       1$              ;Go repeat the process
43         ;
44         ; Finished with this queue
45         ;
46 005714 9$:    ENABL                      ;** Enable interrupts **
47 005722 012605      MOV      (SP)+,R5
48 005724 012603      MOV      (SP)+,R3
49 005726 012602      MOV      (SP)+,R2
50 005730 000207      RETURN

```

MOVQ -- Move queue element to internal queue

```

1          .SBTTL  MOVQ  -- Move queue element to internal queue
2          ;-----
3          ; MOVQ is called to move the current queue element
4          ; onto an internal queue.
5          ;
6          ; Inputs:
7          ; R3 = Address of internal queue header
8          ; R4 = Address of current queue element
9          ; R5 = CL unit index number
10         ;
11 005732 010346
12 005734 010446
13         ;
14         ; Set up R3 to point to queue header but make it look like we are
15         ; pointing to a queue element.
16         ;
17 005736 060503          ADD      R5,R3          ;Point to correct queue head entry
18 005740 162703 000000C  SUB      #Q.LINK-Q.BLKN,R3;Make it look like pointer to a Q element
19         ;
20         ; Add queue entry to tail of internal list
21         ;
22 005744 010400          MOV      R4,R0          ;Save address of new queue element
23 005746          DISABL          ;;; ** Disable interrupts **
24 005754 010304 1$:      MOV      R3,R4          ;;; Remember current queue element address
25 005756 016303 000000C  MOV      Q.LINK-Q.BLKN(R3),R3; ;Get address of next queue element
26 005762 001374          BNE      1$            ;;; Loop till end of list found
27 005764 010064 000000C  MOV      R0,Q.LINK-Q.BLKN(R4); ;Add new entry to end of list
28 005770 005060 000000C  CLR      Q.LINK-Q.BLKN(R0); ;Say it is the end of the list
29         ;
30         ; Finished
31         ;
32 005774          9$:      ENABL          ;;; Enable interrupts **
33 006002 012604          MOV      (SP)+,R4
34 006004 012603          MOV      (SP)+,R3
35 006006 000207          RETURN

```

RTNQ -- Return completed queue elements to the system

```

1          .SBTTL  RTNQ  -- Return completed queue elements to the system
2          -----
3          ; RTNQ is called to return completed queue elements to the system.
4          ;
5          ; Inputs:
6          ;   CQH = Pointer to 1st queue element on list of completed queue elements.
7          ;
8 006010    010446
9 006012    010546
10
11          ; See if this routine is currently being used by someone else.
12          ; If so, just exit. The other user will return all pending queue elements.
13          ;
14 006014    005237    000002'
15 006020    001072
16          ;
17          ; No one else is currently in this routine.
18          ; See if the handler is currently being held.
19          ;
20 006022
21 006030    005737    000000G
22 006034    002023
23          ;
24          ; Handler is being held.
25          ; This means an I/O abort is being done for the handler.
26          ; We cannot return queue elements to the system now.
27          ; Queue a fork request at a low priority which will be held until the
28          ; I/O abort operation is completed.
29          ;
30 006036    005737    000006'
31 006042    001061
32 006044    005237    000006'
33 006050
34 006056    004737    000000G
35 006062    112764    177777G 000000G
36 006070    012764    006010' 000000G
37 006076    004737    000000G
38 006102    000441
39          ;
40          ; This handler is not being held.
41          ; Remove completed queue element from completion list and place it as
42          ; the current queue element for this handler.
43          ;
44 006104    005037    000006'
45 006110    013704    000004'
46 006114    001434
47 006116    016437    000000C 000004'
48 006124    013746    000000G
49 006130    013746    000000G
50 006134    010437    000000G
51 006140    010437    000000G
52 006144    005064    000000C
53          ;
54          ; Now call the system IOFIN routine to release the queue element
55          ;
56 006150
57 006156    012704    000000G

```

```

RTNQ:      MOV      R4,-(SP)
           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.
           ;
           INC      RTNCNT          ;Is someone else already in this routine?
           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.
           ;
           DISABL          ;** Disable interrupts **
           TST      CLABF          ;;; Is handler currently being held?
           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.
           ;
           TST      ABTQFL          ;;; Have we already queued a fork request?
           BNE      3$              ;;; Br if yes
           INC      ABTQFL          ;;; Set flag saying abort fork request queued
           ENABL          ;** Enable interrupts **
           CALL     FRKGET          ;Get a free fork request block
           MOVB     #<FP$IOA-1>,FQ$PRI(R4);Set priority below I/O abort
           MOV      #RTNQ,FQ$RTN(R4);Set address of routine to be called by fork
           CALL     FORKQ          ;Queue the fork request
           BR       3$              ;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.
           ;
6$:      CLR      ABTQFL          ;;; Say abort fork request no longer queued
5$:      MOV      CQH,R4          ;;; Get addr of 1st queue element on compl list
           BEQ      3$              ;;; Br if no more entries to free
           MOV      Q.LINK-Q.BLKN(R4),CQH ;;; Remove entry from completion list
           MOV      CLCQE,-(SP)      ;;; Save current queue element pointer
           MOV      CLLQE,-(SP)      ;;; Also save last queue element pointer
           MOV      R4,CLCQE          ;;; Set entry being freed as current Q element
           MOV      R4,CLLQE          ;;; And as last queue element
           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
           ;
4$:      ENABL          ;** Enable interrupts **
           MOV      #CLCQE,R4          ;Point to CQE cell for IOFIN

```

RTNQ -- Return completed queue elements to the system

```
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
```

LINON -- Turn on a communications line

```

1          .SBTTL  LINON  -- Turn on a communications line
2          ;-----
3          ; LINON is called to turn on a communications line the first time I/O
4          ; is done to the line.
5          ;
6          ; Inputs:
7          ;   R5 = CL unit index number.
8          ;
9 006226    LINON:
10         ;
11         ; Set flag saying line is turned on
12         ;
13 006226    052765    000000G 000000G          BIS      #CM$ON,CL$STA(R5)          ; Say line is turned on
14         ;
15         ; Assert Data Terminal Ready
16         ;
17 006234    052765    000000G 000000G          BIS      #CD$DTR,CL$OPT(R5)          ; Say we want DTR on
18 006242    004737    006250'          CALL      SETDTR          ; Raise the DTR line
19         ;
20         ; Finished
21         ;
22 006246    000207          RETURN

```

SETDTR -- Set Data Terminal Ready status

```

1          .SBTTL  SETDTR -- Set Data Terminal Ready status
2          ;-----
3          ; SETDTR is called to confirm that the Data Terminal Ready status is
4          ; in agreement with the desired state as specified by the CO$DTR
5          ; flag in the unit option flag word (CL$OPT(R5)).
6          ;
7          ; Inputs:
8          ;   R5 = CL unit number index
9          ;
10         SETDTR: MOV     R1, -(SP)
11                MOV     R2, -(SP)
12          ;
13          ; See if he wants DTR on or off
14          ;
15         006250 010146          BIT     #CO$DTR, CL$OPT(R5); Is DTR wanted on or off?
16         006252 010246          BEQ     1$              ; Br if wanted off
17          ;
18          ; DTR is wanted on. See if it is currently on.
19          ;
20         006264 032765 000000G 000000G          BIT     #CM$DTR, CL$STA(R5)      ; Is DTR currently asserted?
21         006272 001022          BNE     9$              ; Br if yes -- all is ok
22         006274 052765 000000G 000000G          BIS     #CM$DTR, CL$STA(R5)      ; Say we are raising DTR
23         006302 012700 000000G          MOV     #MS$DTR, R0          ; Say we want to set DTR
24         006306 000410          BR      2$              ; Go set DTR
25          ;
26          ; DTR is wanted off. See if it is currently off.
27          ;
28         006310 032765 000000G 000000G 1$:          BIT     #CM$DTR, CL$STA(R5)      ; Is DTR currently off?
29         006316 001410          BEQ     9$              ; Br if yes -- all is ok
30         006320 042765 000000G 000000G          BIC     #CM$DTR, CL$STA(R5)      ; Say we are dropping DTR
31         006326 005000          CLR      R0              ; Say we want to drop DTR
32          ;
33          ; Call hardware-dependent routine to change DTR status
34          ;
35         006330 016501 000000G          2$:          MOV     CL$LIX(R5), R1          ; Get line # for this CL unit
36         006334 004737 000000G          CALL     SETDSS          ; Change DTR status
37          ;
38          ; Finished
39          ;
40         006340 012602          9$:          MOV     (SP)+, R2
41         006342 012601          MOV     (SP)+, R1
42         006344 000207          RETURN

```

SETBRK -- Control break transmission

```

1          .SBTTL  SETBRK -- Control break transmission
2          ;-----
3          ; SETBRK is called to start or end transmission of a break character
4          ; to a CL line.
5          ;
6          ; Inputs:
7          ;   R0 = CM$BRK to start sending break; 0 to stop break.
8          ;   R5 = CL unit index number.
9          ;
10         SETBRK: MOV     R1,-(SP)
11         MOV     R2,-(SP)
12         ;
13         ; Call hardware-dependent routine to control break transmission
14         ;
15         MOV     CL$LIX(R5),R1 ;Get line # for this CL unit
16         MOV     LCDTYP(R1),R2 ;Get line control type code
17         OCALL   BRKJMP        ;Call hardware control routine
18         ;
19         ; Finished
20         ;
21         MOV     (SP)+,R2
22         MOV     (SP)+,R1
23         RETURN
24         ;
25         ; Dummy routine used as a jump off point to the CDSBRK routine.
26         ; This is done so that we can use an OCALL to save our overlay number.
27         ;
28         BRKJMP: JMP     @CDSBRK(R2) ;Call hardware routine to control break
29         .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

%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										
CDSDDS	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					

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												
CLSFCD	1-54												
CLSFCDL	1-53												
CLSFHS	1-53												
CLSFIC	1-45												
CLSFMS	1-49												
CLSFOD	1-45												
CLSFOD	1-53	17-76	17-91										
CLSFRL	1-50	19-22											
CLSFSL	1-54												
CLSFSD	1-54												
CLSFSP	1-57												
CLSFSS	1-54												
CLSFSS	1-54												
CLSFSS	1-54												
CLSFSS	1-54												
CLSFSS	1-54												
CLSFSS	1-54												
CLSFSS	1-54												
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\$EOD	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						

[illegible]

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*									
MAXSFO	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#								
QVRHC	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#	
SFGRPO	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	

