

2-	1	GTSLCH	-- Get next character from single-line editor
3-	1	SLINCH	-- See if any input characters need to be processed
4-	1	SLINIT	-- Initialize for new get line
6-	1	ORDCHR	-- Process ordinary characters
7-	1	REGCHR	-- Process normal characters
8-	1	CKUAC	-- Check for user-defined activation characters
9-	1	CKRDTM	-- Check for read time-out character
10-	1	CSCHK	-- Check for start of control sequence
11-	1	EPCH1	-- Accrue a terminal control sequence
12-	1	CSFIN	-- Process terminal control sequence
14-	1	TCUP	-- Up-arrow processing
15-	1	TCDOWN	-- Down-arrow processing
16-	1	TCLEFT	-- Left-arrow processing
17-	1	TCRITE	-- Right-arrow processing
18-	1	TCPF1	-- PF1 processing
19-	1	TCPF2	-- PF2 processing
20-	1	TCPF3	-- PF3 processing
21-	1	TCPF4	-- PF4 processing
22-	1	TCENTR	-- Enter key processing
22-	19	TCINVL	-- Invalid function key
23-	1	KEY0	-- Key "0" processing
24-	1	KEY1	-- Key "1" processing
25-	1	KEY2	-- Key "2" processing
26-	1	KEY3	-- Key "3" processing
27-	1	KEY4	-- Key "4" processing
28-	1	KEY5	-- Key "5" processing
29-	1	KEY6	-- Key "6" processing
30-	1	KEY7	-- Key "7" processing
31-	1	KEY8	-- Key "8" processing
32-	1	KEY9	-- Key "9" processing
33-	1	KEYCOM	-- Key ",", " processing
34-	1	KEYMIN	-- Key "--" processing
35-	1	KBS	-- Save a command
36-	1	KBX	-- Recall the saved command
37-	1	E1	-- E1 processing
38-	1	E2	-- E2 processing
39-	1	E3	-- E3 processing
40-	1	E4	-- E4 processing
41-	1	E5	-- E5 processing
42-	1	E6	-- E6 processing
43-	1	F6	-- F6 processing
44-	1	F7	-- F7 processing
45-	1	F8	-- F8 processing
46-	1	F9	-- F9 processing
47-	1	F10	-- F10 processing
48-	1	F11	-- F11 processing
49-	1	F12	-- F12 processing
50-	1	F13	-- F13 processing
51-	1	F14	-- F14 processing
52-	1	F15	-- F15 processing
53-	1	F16	-- F16 processing
54-	1	F17	-- F17 processing
55-	1	F18	-- F18 processing
56-	1	F19	-- F19 processing
57-	1	F20	-- F20 processing
58-	1	KEYDOT	-- Key "." processing
59-	1	DOCTRL	-- Process control characters

60-	1	ICPCR	-- Carriage-return processing
61-	1	ICPLF	-- Line-feed processing
62-	1	ICPBS	-- Backspace processing
63-	1	ICPESC	-- Escape processing
64-	1	ICPCTA	-- Control-A processing
65-	1	ICPCTC	-- Control-C processing
66-	1	ICPCTR	-- Control-R processing
67-	1	ICPCTU	-- Control-U processing
68-	1	ICPCTZ	-- Control-Z processing
69-	1	ICPRUB	-- Rubout processing
70-	1	ICPNUL	-- Null processing
71-	1	INWAIT	-- Wait for input characters
72-	1	SAVLIN	-- Save current input line
73-	1	LINFIN	-- Terminate input line
74-	1	DELLFT	-- Delete character to left of cursor
75-	1	SLMVUP	-- Move up to previous stored command
76-	1	SLMVDN	-- Move down to previous stored command
77-	1	RSTLIN	-- Restore a full line
79-	1	OSTRIK	-- Overstrike
80-	1	INSERT	-- Insert string into edit buffer
81-	1	PAINT	-- Redisplay current edit line
82-	1	MAXLEN	-- Compute maximum allowed line length
83-	1	COLCLC	-- Calculate column position of a character
84-	1	CHRPOS	-- Move cursor to a specific character
85-	1	CURPOS	-- Move cursor to a specified column
86-	1	CSRRIT	-- Generate control sequence to move cursor right
87-	1	CSRLFT	-- Generate control sequence to move cursor left
88-	1	CHKDLM	-- Check for word delimiters
89-	1	CVTLC	-- Convert lower-case characters to upper-case
90-	1	ECOCTL	-- Echo a control character
90-	27	RNGBEL	-- Ring bell to signal error
90-	39	AKEYON	-- Turn on alternate keypad mode
91-	1	ECHO	-- Send character to the terminal
92-	1	KEYCK	-- Check to see if key is defined
93-	1	KEYSRC	-- See if terminal key is defined by user
94-	1	KEYSUB	-- Substitute a defined string for a key
95-	1	CHK52	-- Determine if terminal is a VT52

```

1          .TITLE  TSSLE -- TSX-Plus Single Line Editor
2          .ENABL  LC
3          .ENABL  AMA
4          .DSABL  GBL
5          .CSECT  TSSLE
6 000000    074245  TSSLE:  .RAD50  /SLE/          ;Overlay region id
7          ;
8          ; TSSLE is the TSX-Plus system overlay module that contains routines
9          ; to implement the single line editor.
10         ;
11         ; Copyright (c) 1983, 1984, 1985, 1986, 1987, 1988.
12         ; S&H Computer Systems, Inc.
13         ; Nashville, Tennessee USA
14         ; All rights reserved.
15         ;
16         ; Global definitions
17         ;
18         .GLOBL  QTSLECH
19         ;
20         ; Global references
21         ;
22         .GLOBL  SLOVER, SLRPTR, SLCYC1, SLCYC2, $scca, lsw5
23         .GLOBL  $VBELL, $PWKEY, LSW11, LNPRIM, $V52EM, SLSPTR, SLDOWN
24         .GLOBL  VVLSCH, $CTRLW, MAXSEC, DOSWIT, $1ESC, $WDISP, WINDSP
25         .GLOBL  KT$LET, KT$GLT, $SLLET, LWINDO, VVPWCH, WINPRT
26         .GLOBL  VKEYMX, KD$COD, KD$FLG, KD$TXT, KD$$SZ, KF$EOD, KF$TRM
27         .GLOBL  KT$NRM, KT$GLD, KC$E1, KC$E2, KC$E3, KC$E4, KC$E5, KC$E6
28         .GLOBL  KC$F6, KC$F7, KC$F8, KC$F9, KC$F10, KC$F11, KC$F12, KC$F13
29         .GLOBL  KC$F14, KC$F15, KC$F16, KC$F17, KC$F18, KC$F19, KC$F20
30         .GLOBL  KC$MIN, KC$ENT, KC$COM, KC$DOT, KC$PF1, KC$PF2, KC$PF3, KC$PF4
31         .GLOBL  KC$KP0, KC$KP1, KC$KP2, KC$KP3, KC$KP4, KC$KP5, KC$KP6
32         .GLOBL  KC$KP7, KC$KP8, KC$KP9, KC$LFT, KC$RIT, KC$UP, KC$DWN, VPAR6
33         .GLOBL  SLCBUF, SIGWAT, LSW6, CTRLQ, LF, LFWLIM, LTRMTP, KEYPAR
34         .GLOBL  LOGFLG, LF$IN, SLLBUF, LRBFIL, LCOL, INTPRI, LHIPCT
35         .GLOBL  SLOPTR, SLECOL, LSPACT, BKSPAC, SLGX, VT52, TAB, $XSTOP
36         .GLOBL  SLSCOL, SLCCOL, PR7, LSW3, STOP, LTTPAR, $NOIN, LOGCR, CHKABT
37         .GLOBL  SLGOLD, VINTIO, LACTIV, QHDSFN, KPAR6, CR, PSW
38         .GLOBL  LOGCHR, LNSPAC, S$INWT, SLEBUF, SLMXLN, BELL, LINPNT
39         .GLOBL  OVRHC, $DBGMD, SLDBUF, LAFSIZ, DELCHR, $DISCN
40         .GLOBL  ESC, QUECHR, $NOLF, $LC, LSW2, LCBIT, LJSW, $SLINI, LSW7
41         .GLOBL  LSCCA, SLCR, $ECHO, LRTCHR, $SLKED, SLBACK, SLSBUF, LINCNT
42         .GLOBL  VT100, LITIME, VQUAN1, LSTATE, S$TTFN, ENQTL, $SUCF, LSW9
43         .GLOBL  $NOINT, SLLPTR, SLEND, RUBOUT, $NDICP, LSW10
44         .GLOBL  CSICHR, SS3CHR, SLCSBF, SLCSBX, SLCSPT, SLCSR, NEDCDI
45         .GLOBL  $DETCH, LSW
46         ;
47         ; Macro definitions:
48         ;
49         .MACRO  DISABL          ;DISABLE INTERRUPTS
50         BIS     #PR7, @#PSW
51         .ENDM   DISABL
52
53         .MACRO  ENABL          ;ENABLE INTERRUPTS
54         BIC     INTPRI, @#PSW
55         .ENDM   ENABL
56
57         .MACRO  OCALL  ENTADD

```

```

58      .IF      B,ENTADD
59      .ERROR   ;OCALL without entry address
60      .ENDC
61      CALL     OVRHC
62      .WORD    ENTADD
63      .ENDM    OCALL
64
65      ; The TTMAP and TTMAPX macros are used to map kernel-mode par6 to the
66      ; terminal character buffer area. The previous contents of par6 map
67      ; register are pushed on the stack and may be restored by using the
68      ; UNMAP or UNMAPX macros.
69      ; R1 must contain the line index number of the line whose buffers
70      ; are being accessed.
71      ; The difference between the TTMAP-UNMAP macros and the TTMAPX-UNMAPX
72      ; macros is that the X-versions are more efficient but may only be
73      ; used from within interrupt service routines where we are guaranteed
74      ; to be running on the system stack.
75      ; The TTMAP and UNMAP versions of the macros must only be
76      ; used in sections of code where the interrupts are disabled.
77
78      .MACRO    TTMAPX
79      MOV       LTPAR(R1),@#KPAR6
80      .ENDM    TTMAPX
81
82      .MACRO    UNMAPX
83      .ENDM    UNMAPX
84
85      .MACRO    TTMAP
86      MOV       @KPAR6,MAPHLD
87      MOV       LTPAR(R1),@#KPAR6
88      .ENDM    TTMAP
89
90      .MACRO    UNMAP
91      MOV       MAPHLD,@#KPAR6
92      .ENDM    UNMAP
93
94      ; The KEYMAP macro is used to map KPAR6 to the local region that has
95      ; the user key definitions. Use the KEYUMP macro to restore mapping.
96
97      .MACRO    KEYMAP
98      MOV       @KPAR6,MAPHLD
99      DISABL
100      MOV       KEYPAR,@#KPAR6
101      .ENDM    KEYMAP
102
103      .MACRO    KEYUMP
104      MOV       MAPHLD,@#KPAR6
105      ENABL
106      .ENDM    KEYUMP
107
108      ; The KEYCHK macro is used to call the KEYSRC routine to see if a specific
109      ; key has been defined by the user.
110      ; The argument to KEYCHK is the key code.
111
112      .MACRO    KEYCHK KEYCOD
113      JSR       R5,KEYCK
114      .WORD     KEYCOD

```

```
115          .ENDM  KEYCHK
116          ;
117          ; Data areas
118          ;
119 000002 000000  MAPHLD: .WORD 0          ;Temp cell used by TTMAP macro
```

GTSLCH -- Get next character from single-line editor

```

1          .SBTTL  GTSLCH -- Get next character from single-line editor
2          ;-----
3          ; GTSLCH is called to get from the single-line editor the next character
4          ; to pass to the program.
5          ;
6          ; Inputs:
7          ;   R1 = Job index number.
8          ;
9          ; Outputs:
10         ;   R0 = Next character to pass to program.
11         ;
12 000004 010246 GTSLCH: MOV      R2,-(SP)
13         ;
14         ; Perform initialization if this is the first call for this line
15         ;
16 000006 032761 000000G 000000G          BIT      ##SLINI,LSW7(R1);Have we done initialization for this line?
17 000014 001002          BNE      1$          ;Br if yes
18 000016 004737 000176'          CALL     SLINIT          ;Initialize for line
19         ;
20         ; See if we have a complete line ready yet.
21         ;
22 000022 013702 000000G          1$:      MOV      SLOPTR,R2          ;Do we have any characters ready to go?
23 000026 001404          BEQ      2$          ;Br if not
24         ;
25         ; Get next output character
26         ;
27 000030 112200          MOVB     (R2)+,R0          ;Get next output character
28 000032 001005          BNE      3$          ;Br if got a character
29         ;
30         ; We have finished passing the last line to the program.
31         ; Initialize for a new line.
32         ;
33 000034 004737 000176'          CALL     SLINIT          ;Initialize for a new line
34         ;
35         ; There are no characters available to be passed to program.
36         ; See if there are any pending input characters that need
37         ; to be processed.
38         ;
39 000040 004737 000102'          2$:      CALL     SLINCH          ;Check for pending input characters
40         ;
41         ; Now go back and see if we have any characters ready for output
42         ;
43 000044 000766          BR       1$
44         ;
45         ; We might have a character for the program
46         ;
47 000046 010237 000000G          3$:      MOV      R2,SLOPTR          ;Save updated output character pointer
48         ;
49         ; See if we need to reset the "interactive" timer for this job
50         ;
51 000052 032761 000000G 000000G          5$:      BIT      ##NOINT,LSW7(R1) ;Is program being run non-interactively?
52 000060 001006          BNE      4$          ;Br if yes
53 000062 013761 000000G 000000G          MOV      VINTIO,LHIPCT(R1) ;Reset high-priority hit limit for job
54 000070 013761 000000G 000000G          MOV      VQUAN1,LITIME(R1) ;Reset interactive CPU time limit
55         ;
56         ; Finished
57         ;

```

TSSLE -- TSX-Plus Single Line E MACRO V05.05 Wednesday 18-Jan-89 14:16 Page 2-1
GTSLCH -- Get next character from single-line editor

58 000076 012602
59 000100 000207

4\$: MOV (SP)+, R2
RETURN

```

1          .SBTTL  SLINCH -- See if any input characters need to be processed
2          ;-----
3          ; SLINCH is called to see if there are any characters pending in the
4          ; terminal input buffer that need to be processed by the single
5          ; line editor.
6          ; This routine waits for an input character to be received, processes it,
7          ; and then returns.
8          ;
9          ; Inputs:
10         ; R1 = Line index number.
11         ;
12 000102 010246 SLINCH: MOV      R2,-(SP)
13 000104 010546      MOV      R5,-(SP)
14         ;
15         ; Abort the job if a detached job is trying to do input from the terminal
16         ;
17 000106 032761 000000G 000000G      BIT      ##DETCH,LSW(R1) ; Is this a detached job?
18 000114 001405      BEQ      1$      ; Br if not
19 000116 052761 000000G 000000G      BIS      ##DISCN,LSW(R1) ; Abort the detached job
20 000124 004737 000000G      CALL     STOP      ; Stop execution of job
21         ;
22         ; See if there are any characters in the terminal input buffer
23         ;
24 000130 005761 000000G 1$: TST      LINCNT(R1) ; Are there any chars in input buffer?
25 000134 001003      BNE      2$      ; Br if yes
26         ;
27         ; There are no pending characters.
28         ; Wait until we get one.
29         ;
30 000136 004737 007042'      CALL     INWAIT    ; Wait for a character to arrive
31 000142 000772      BR       1$      ; Go try again
32         ;
33         ; There are pending input characters.
34         ; Get the next character out of the input buffer.
35         ;
36 000144 016102 000000G 2$: MOV      LIMPNT(R1),R2 ; Get address of next character to get
37 000150      OCALL     DELCHR      ; Get character from buffer
38 000156 042700 177400      BIC      #^C377,R0 ; Clear all but low order 8 bits of character
39 000162 010005      MOV      R0,R5      ; Put character we got in R5
40         ;
41         ; Process the character
42         ;
43 000164 004737 000350'      CALL     PRCHAR    ; Process the character
44         ;
45         ; Return to higher-level routine to see if we have a complete
46         ; line ready yet.
47         ;
48 000170 012605      MOV      (SP)+,R5
49 000172 012602      MOV      (SP)+,R2
50 000174 000207      RETURN

```

SLINIT -- Initialize for new get line

```

1          .SBTTL  SLINIT -- Initialize for new get line
2          -----
3          ; SLINIT is called each time we begin to acquire a new input line.
4          ;
5          ; Inputs:
6          ;   R1 = Job index number.
7          ;
8 000176 010546 SLINIT: MOV      R5, -(SP)
9          ;
10         ; If SL is in KED mode, send control sequence to terminal to put
11         ; numeric keypad into alternate mode.
12         ;
13 000200 032761 000000G 000000G      BIT      $$SLKED, LSW7(R1); Are we in KED mode?
14 000206 001402      BEQ      1$      ; Br if not
15 000210 004737 011166'      CALL     AKEYON      ; Turn on alternate keypad mode
16         ;
17         ; Say no line is ready to be sent to program
18         ;
19 000214 005037 000000G 1$:      CLR      SLOPTR      ; No output line ready
20         ;
21         ; Say Gold key (PF1) is not pending
22         ;
23 000220 105037 000000G      CLRB     SLGOLD      ; Gold key not pending
24         ;
25         ; Say carriage return not pending
26         ;
27 000224 105037 000000G      CLRB     SLCR      ; Carriage return was not last character
28         ;
29         ; Say we are not processing a terminal control sequence
30         ;
31 000230 005037 000000G      CLR      SLCSR      ; Not processing terminal control sequence
32         ;
33         ; Set direction to forward
34         ;
35 000234 105037 000000G      CLRB     SLBACK     ; Set direction = forward
36         ;
37         ; Say we are not in overstrike mode
38         ;
39 000240 105037 000000G      CLRB     SLOVER     ; Not in overstrike mode
40         ;
41         ; Clear edit buffer and initialize cursor to point to 1st character
42         ;
43 000244 105037 000000G      CLRB     SLEBUF     ; Store null as 1st char in edit buffer
44 000250 012737 000000G 000000G      MOV     #SLEBUF, SLCX      ; Say cursor is pointing to 1st character
45         ;
46         ; Set up cursor display position for left end of line
47         ;
48 000256 116100 000000G      MOVB     LCOL(R1), R0      ; Get column number of start of field
49 000262 010037 000000G      MOV      R0, SLSCOL     ; Set this as start-of-line column
50 000266 010037 000000G      MOV      R0, SLECOL     ; Set this as end-of-line column
51 000272 010037 000000G      MOV      R0, SLCCOL     ; Set as cursor display column
52         ;
53         ; Initialize to point to most recent saved line
54         ;
55 000276 005737 000000G      TST      SLSPTR     ; Has save-line pointer been initialized?
56 000302 001003      BNE      2$      ; Br if yes
57 000304 012737 000000G 000000G      MOV     #SLLBUF, SLSPTR ; Most recently saved line position

```

```
58 000312 013737 000000G 000000G 2$:      MOV      SLSPTR,SLLPTR      ;Set up-arrow pointer to most recent line
59                                           ;
60                                           ; Set flag saying initialization has been done for this line
61                                           ;
62 000320 052761 000000G 000000G          BIS      $$SLINI,LSW7(R1);Set initialization-done flag
63                                           ;
64                                           ; See if we need to recall a line
65                                           ;
66 000326 013705 000000G          MOV      SLRPTR,R5          ;Is a recall pending?
67 000332 001404          BEQ      9$          ;Br if not
68 000334 004737 007704'          CALL     RSTLIN          ;Restore the line
69 000340 005037 000000G          CLR      SLRPTR          ;Say recall has been done
70                                           ;
71                                           ; Finished
72                                           ;
73 000344 012605          9$:      MOV      (SP)+,R5
74 000346 000207          RETURN
```

SLINIT -- Initialize for new get line

```

1
2 ; -----
3 ; Process a character received from the terminal.
4 ;
5 ; Inputs:
6 ;   R1 = Job's virtual line index number.
7 ;   R5 = Received character.
8 ;
9 PRCHAR: MOV     R2, -(SP)
10          MOV    R3, -(SP)
11          MOV    R5, -(SP)
12 ;
13 ; Determine if this character begins a terminal control sequence
14 ;
15          CALL    CSCHK           ; Check for control sequence start
16          BCC     9$             ; Br if start of control sequence
17 ;
18 ; See if we are currently accepting a terminal control sequence
19 ;
20          MOV     SLCSR, R0       ; Are we processing a control sequence?
21          BEQ     3$             ; Br if not
22          CALL    (R0)           ; Call routine to process this character
23          BR      9$             ; Finished with this character
24 ;
25 ; Determine if this is a user-defined key
26 ;
27 3$:      TST     KEYPAR          ; Are there any user-defined keys?
28          BEQ     5$             ; Br if not
29          MOV     R5, R0         ; Get the character
30          TSTB    SLGOLD         ; Was gold key pressed?
31          BEQ     8$             ; Br if not
32          BIS     #KT$GLT*400, R0 ; Say this is a gold letter
33          BR      2$
34 8$:      BIS     #KT$LET*400, R0 ; Say this is a regular letter
35 2$:      CALL    KEYSRC         ; See if key is user defined
36          BCS     9$             ; Br if this is a user-defined key
37 ;
38 ; Process ordinary characters that are not part of control sequences
39 ; and that are not user-defined keys.
40 ;
41 5$:      CALL    ORDCHR         ; Process an ordinary character
42 ;
43 ; Finished
44 ;
45 9$:      MOV     (SP)+, R5
46          MOV     (SP)+, R3
47          MOV     (SP)+, R2
48          RETURN

```

ORDCHR -- Process ordinary characters

```

1          .SBTTL  ORDCHR -- Process ordinary characters
2          ;-----
3          ; Process ordinary characters (both control and non-control).
4          ;
5          ; Inputs:
6          ;   R1 = Job index number
7          ;   R5 = Received character
8          ;
9 000450   ORDCHR:
10          ;
11          ; See if Gold key was pressed
12          ;
13 000450   105737   000000G   TSTB    SLGOLD    ; Was gold key pressed?
14 000454   001426   BEQ      1$      ; Br if not
15          ;
16          ; Gold key was pressed. Check for Gold-S.
17          ;
18 000456   120527   000123   CMPB    R5,#'S    ; Gold-S?
19 000462   001403   BEQ      2$      ; Br if yes
20 000464   120527   000163   CMPB    R5,#'s    ; Gold-s?
21 000470   001005   BNE      3$      ; Br if not
22 000472   105037   000000G   2$: CLRB    SLGOLD    ; Reset Gold flag
23 000476   004737   004462'   CALL    KBS      ; Gold-S ==> Save command
24 000502   000443   BR       9$
25          ;
26          ; Check for Gold-X.
27          ;
28 000504   120527   000130   3$: CMPB    R5,#'X    ; Gold-X?
29 000510   001403   BEQ      10$     ; Br if yes
30 000512   120527   000170   CMPB    R5,#'x    ; Gold-x?
31 000516   001005   BNE      1$      ; Br if not
32 000520   105037   000000G   10$: CLRB   SLGOLD    ; Reset Gold flag
33 000524   004737   004514'   CALL    KBX      ; Gold-X ==> Recall command
34 000530   000430   BR       9$
35          ;
36          ; If debugger is in control, bypass some special character processing
37          ;
38 000532   032761   000000G 000000G 1$: BIT    ##DBGMD,LSW6(R1); Is debugging program in control?
39 000540   001006   BNE      6$      ; If yes then bypass some checking
40          ;
41          ; See if this is a user-defined activation character
42          ;
43 000542   004737   000774'   CALL    CKUAC    ; See if this is a user-defined activation char
44 000546   103021   BCC      9$      ; Br if it is a user-defined activation char
45          ;
46          ; See if this is a read-time activation character
47          ;
48 000550   004737   001074'   CALL    CKRDTM    ; See if this is a read time-out character
49 000554   103016   BCC      9$      ; Br if it is
50          ;
51          ; Determine if this is a normal or control character
52          ;
53 000556   020527   000037   6$: CMP     R5,#37    ; Is this a normal or control char?
54 000562   101003   BHI      4$      ; Br if not control character
55 000564   004737   005406'   CALL    DOCTRL    ; Process a control character
56 000570   000410   BR       9$
57 000572   120527   000000G   4$: CMPB    R5,#RUBOUT ; Is this a rubout character?

```

ORDCHR -- Process ordinary characters

```
58 000576 001003          BNE      7$          ;Br if not
59 000600 004737 006764'   CALL     ICPRUB      ;Process rubout character
60 000604 000402          BR       9$
61 000606 004737 000614'   7$:    CALL     REGCHR      ;Process a normal character
62                          ;
63                          ; Finished
64                          ;
65 000612 000207          9$:    RETURN
```

REGCHR -- Process normal characters

```

1          .SBTTL  REGCHR -- Process normal characters
2          ;-----
3          ; Process normal (non-control) characters.
4          ;
5          ; Inputs:
6          ;   R1 = Virtual line index number.
7          ;   R5 = Current input character.
8          ;
9          REGCHR: MOV     R2,-(SP)
10         MOV     R3,-(SP)
11         MOV     R4,-(SP)
12         MOV     R5,-(SP)
13         ;
14         ; See if this is part of a request to switch to a virtual line
15         ;
16         000624 032761 000000G 000000G      BIT     $$CTRLW,LSW3(R1); Was ctrl-W the last character?
17         000632 001427                      BEQ     1$          ; Br if not
18         000634 042761 000000G 000000G      BIC     $$CTRLW,LSW3(R1); Say ctrl-W no longer the last char
19         000642 162705 000060                SUB     #'0,R5          ; Convert digit to binary value
20         000646 002445                      BLT     9$          ; Br if char after ctrl-W not a digit
21         000650 020527 000000G                CMP     R5,$MAXSEC      ; Don't exceed max line # allowed
22         000654 003042                      BGT     9$          ; Br if too big
23         000656                      OCALL    DOSWIT          ; Switch to a virtual line
24         000664 052761 000000G 000000G      BIS     $$VBELL,LSW9(R1); Suppress bell ring on next SL read
25         000672 032761 000000G 000000G      BIT     $$WDISP,LSW6(R1); Do we need to redisplay our window?
26         000700 001430                      BEQ     9$          ; Br if not
27         000702                      OCALL    WINDSP          ; Redisplay window for job
28         000710 000424                      BR      9$
29         ;
30         ; Translate character to upper case
31         ;
32         000712 110500                      1$:  MOVB    R5,R0          ; Get character
33         000714 004737 011066'                CALL    CVTLC          ; Convert lower-case to upper-case
34         ;
35         ; Insert character into buffer
36         ;
37         000720 105737 000000G                TSTB    SLOVER          ; Are we in overstrike mode?
38         000724 001403                      BEQ     2$          ; Br if not
39         000726 004737 010066'                CALL    OSTRIK          ; Overstrike
40         000732 000405                      BR      3$
41         000734 012702 000000G      2$:  MOV     $SLCBUF,R2          ; Point to character buffer
42         000740 110012                      MOVB    R0,(R2)          ; Store char into insert buffer
43         000742 004737 010140'                CALL    INSERT          ; Insert character to left of cursor
44         ;
45         ; Say there is no deleted character being held and not in terminal
46         ; control sequence.
47         ;
48         000746 105037 000000G      3$:  CLRB    SLCBUF          ; No deleted character being held
49         000752 005037 000000G                CLR     SLCSR          ; We are not in a terminal control sequence
50         000756 105037 000000G                CLRB    SLGOLD          ; Gold key (PF1) has not been pressed
51         ;
52         ; Finished
53         ;
54         000762 012605      9$:  MOV     (SP)+,R5
55         000764 012604                MOV     (SP)+,R4
56         000766 012603                MOV     (SP)+,R3
57         000770 012602                MOV     (SP)+,R2

```

TSSLE -- TSX-Plus Single Line E MACRO V05.05 Wednesday 18-Jan-89 14:16 Page 7-1
REGCHR -- Process normal characters

58 000772 000207

RETURN

```

1          .SBTTL  CKUAC  -- Check for user-defined activation characters
2          ;-----
3          ; CKUAC is called to determine if the current input character is a
4          ; user-defined activation character.  If it is, it is processed here.
5          ;
6          ; Inputs:
7          ;   R1 = Job index number.
8          ;   R5 = Current input character
9          ;
10         ; Outputs:
11         ;   C-flag cleared ==> This is a user-defined activation character.
12         ;   C-flag set    ==> This is not a user-defined activation char.
13         ;
14 000774 010246  CKUAC:  MOV     R2,-(SP)
15 000776 010346      MOV     R3,-(SP)
16         ;
17         ; See if there are any user-defined activation characters
18         ;
19 001000 016102 0000000  MOV     LNSPAC(R1),R2  ;Get number of user-defined activation chars
20 001004 001427      BEQ     B$                ;Br if there are none
21         ;
22         ; There are some user-defined activation characters.
23         ; Translate input character to upper-case if that is wanted.
24         ;
25 001006 010500      MOV     R5,R0                ;Get current character
26 001010 004737 011066'  CALL    CVTLC          ;Convert to upper-case if needed
27         ;
28         ; Search for input character in table of activation characters.
29         ;
30 001014 016103 0000000  MOV     LSPACT(R1),R3  ;Get pointer to table of activation chars
31 001020 120023 1$:      CMPB    R0,(R3)+        ;Is this an activation character?
32 001022 001402      BEQ     2$                ;Br if yes
33 001024 077203      SOB     R2,1$            ;Loop if more to check
34 001026 000416      BR      B$                ;This is not a user-defined activation char
35         ;
36         ; This character is a user-defined activation character.
37         ; Store character at end of line and signal line-completed.
38         ;
39 001030 010005 2$:      MOV     R0,R5                ;Get converted character to R5
40 001032 004737 007172'  CALL    SAVLIN          ;Save input line before storing activation chr
41 001036 013703 0000000  MOV     SLCX,R3          ;Point to cursor character
42 001042 105723 3$:      TSTB    (R3)+          ;Search for null at end of line
43 001044 001376      BNE     3$                ;Loop till null found
44 001046 005303      DEC     R3                ;Point to the null character
45 001050 110523      MOVB    R5,(R3)+          ;Store activation char as last char in field
46 001052 105023      CLRB    (R3)+          ;Store null at end
47 001054 004737 007346'  CALL    LINFIN          ;Say input line is finished
48 001060 000241      CLC                    ;Say this is an activation character
49 001062 000401      BR      9$
50         ;
51         ; Character is not a user-defined activation character.
52         ;
53 001064 000261 8$:      SEC                    ;Say this is not a user-defined activation chr
54         ;
55         ; Finished
56         ;
57 001066 012603 9$:      MOV     (SP)+,R3

```

TSSLE -- TSX-Plus Single Line E MACRO V05.05 Wednesday 18-Jan-89 14:16 Page 8-1
CKUAC -- Check for user-defined activation characters

58 001070 012602
59 001072 000207

MOV (SP)+, R2
RETURN

CKRDTM -- Check for read time-out character

```

1          .SBTTL  CKRDTM -- Check for read time-out character
2          ;-----
3          ; CKRDTM is called to determine if the current input character is a
4          ; read time-out character.  If it is, the current input line is terminated.
5          ;
6          ; Inputs:
7          ;   R1 = Job index number.
8          ;   R5 = Current input character.
9          ;
10         ; Outputs:
11         ;   C-flag cleared ==> This is a read time-out character.
12         ;
13 001074   CKRDTM:
14         ;
15         ; See if program has specified a read time-out
16         ;
17 001074   016100   000000G      MOV     LRTCHR(R1),R0      ;Is there a read time-out specified?
18 001100   001417      BEQ     B$              ;Br if not
19         ;
20         ; See if current character is time-out character
21         ;
22 001102   120500      CMPB    R5,R0              ;Is current character the time-out char?
23 001104   001015      BNE     B$              ;Br if not
24         ;
25         ; Read time-out has occurred.
26         ; Terminate current input line.
27         ;
28 001106   013700   000000G      MOV     SLCX,R0              ;Get current cursor character index
29 001112   105720 1$:      TSTB    (R0)+          ;Search for end of line
30 001114   001376      BNE     1$              ;Loop till null hit
31 001116   005300      DEC     R0              ;Point to null
32 001120   110520      MOVB    R5,(R0)+        ;Store time-out character at end
33 001122   105010      CLRB    (R0)           ;Put null at end of string
34 001124   004737   007346'      CALL   LINFIN          ;Terminate the input line
35         ;
36         ; Clear the time-out character flag
37         ;
38 001130   005061   000000G      CLR     LRTCHR(R1)        ;Say we have processed the time-out character
39 001134   000241      CLC              ;Say this was the timeout character
40 001136   000401      BR      9$
41         ;
42         ; This is not a read time-out character
43         ;
44 001140   000261  B$:      SEC              ;Signal not time-out character
45         ;
46         ; Finished
47         ;
48 001142   000207  9$:      RETURN

```

CSCHK -- Check for start of control sequence

```

1          .SBTTL  CSCHK  -- Check for start of control sequence
2          ;-----
3          ; CSCHK is called to determine if the current character begins a
4          ; terminal control sequence.
5          ;
6          ; Inputs:
7          ;   R1 = Line index number.
8          ;   R5 = Received character.
9          ;
10         ; Outputs:
11         ;   C-flag cleared ==> Start of control sequence.
12         ;   C-flag set    ==> Not start of control sequence.
13         ;
14 001144  CSCHK:
15         ;
16         ; See if this is the start of an escape sequence
17         ;
18 001144 120527 0000000  CMPB    R5,#ESC      ;Is this character escape?
19 001150 001003          BNE     1$          ;Br if not
20 001152 004737 006274'  CALL    ICPEsc      ;Begin escape sequence
21 001156 000413          BR      8$
22         ;
23         ; See if this character is CSI, beginning of VT200 control sequence
24         ;
25 001160 120527 0000000 1$:  CMPB    R5,#CSICHR    ;CSI character?
26 001164 001003          BNE     2$          ;Br if not
27 001166 004737 006326'  CALL    ICPCSI     ;Start VT200 control sequence
28 001172 000405          BR      8$
29         ;
30         ; See if this character is SS3, beginning of VT200 control sequence
31         ;
32 001174 120527 0000000 2$:  CMPB    R5,#SS3CHR    ;SS3 character?
33 001200 001004          BNE     7$          ;Br if not
34 001202 004737 006352'  CALL    ICPSS3     ;Start VT200 control sequence
35         ;
36         ; We began a control sequence
37         ;
38 001206 000241 8$:      CLC              ;Signal control sequence began
39 001210 000401          BR      9$
40         ;
41         ; We did not begin a control sequence
42         ;
43 001212 000261 7$:      SEC              ;Signal that control sequence did not begin
44         ;
45         ; Finished
46         ;
47 001214 000207 9$:      RETURN

```

```

1          .SBTTL EPCH1 -- Accrue a terminal control sequence
2
3          ; The routines in this section implement a finite state machine
4          ; which accrues the characters of a terminal control sequence.
5          ; While we are accruing a control sequence, the cell SLCSR contains
6          ; the address of the routine to be called to process the next character.
7
8          ; -----
9          ; EPCH1 is called when we receive the first character following escape
10         ; in a VT100 escape sequence.
11
12 001216 013700 000000G EPCH1: MOV SLCSPT,RO ;Get pointer into buffer
13 001222 110520         MOV R5,(RO)+ ;Accrue the character
14 001224 010037 000000G MOV RO,SLCSPT ;Save new pointer
15 001230 012737 001304' 000000G MOV #EPCH2,SLCSR ;Set address of routine for next character
16 001236 000207         RETURN
17
18         ; -----
19         ; EP52 is called when we receive the first character following escape
20         ; in a VT52 escape sequence.
21
22 001240 013700 000000G EP52: MOV SLCSPT,RO ;Get pointer into buffer
23
24         ; If this character is "?" then we are receiving a two character
25         ; control sequence. Otherwise this is a single character control sequence.
26
27 001244 120527 000077         CMPB R5,#'? ;Alternate keypad leadin character?
28 001250 001010         BNE 1$ ;Br if not
29
30         ; This is a two character control sequence.
31         ; Convert "?" to "O" to make the sequence compatible with a VT100.
32
33 001252 112720 000117         MOV R5,#'O,(RO)+ ;Store "O" as 1st char of sequence
34 001256 010037 000000G MOV RO,SLCSPT ;Store new buffer pointer
35 001262 012737 001304' 000000G MOV #EPCH2,SLCSR ;Set address of routine to process more chars
36 001270 000404         BR 9$
37
38         ; This is a single character control sequence
39
40 001272 110520 1$: MOV R5,(RO)+ ;Store character into buffer
41 001274 105020         CLRB (RO)+ ;This terminates the control sequence
42 001276 004737 001356' CALL CSFIN ;End of control sequence
43
44         ; Finished
45
46 001302 000207 9$: RETURN
47
48         ; -----
49         ; EPCH2 process characters after the 1st character of a control sequence.
50
51 001304 013700 000000G EPCH2: MOV SLCSPT,RO ;Get pointer to control sequence buffer
52
53         ; Make sure we are not about to overflow the control sequence buffer
54
55 001310 020027 000000G CMP RO,#SLCSBX ;Is buffer full?
56 001314 103405         BLO 1$ ;Br if not
57 001316 004737 011154' CALL RRGBEL ;Ring the bell

```

EPCH1 -- Accrue a terminal control sequence

```
58 001322 005037 000000G          CLR      SLCSR          ; Say not processing a control sequence
59 001326 000412          BR        9$
60                               ;
61                               ; Store character into control sequence buffer
62                               ;
63 001330 110520          1$:      MOVB     R5,(R0)+          ; Store character into control sequence buffer
64                               ;
65                               ; Determine if this is the terminating character of the control sequence
66                               ;
67 001332 120527 000100          CMPB     R5,#100            ; Is this the final char of control seq?
68 001336 103404          BLO      2$                        ; Br if not
69 001340 105020          CLRB     (R0)+                    ; Terminate the control sequence
70 001342 004737 001356'        CALL     CSFIN              ; Process the control sequence
71 001346 000402          BR        9$
72                               ;
73                               ; This is not the final character of the control sequence.
74                               ; Continue accruing characters.
75                               ;
76 001350 010037 000000G        2$:      MOV      R0,SLCSPT          ; Save new buffer pointer
77                               ;
78                               ; Finished
79                               ;
80 001354 000207          9$:      RETURN
```

CSFIN -- Process terminal control sequence

```

1          .SBTTL  CSFIN  -- Process terminal control sequence
2          ;-----
3          ; CSFIN is called when we have finished accruing a terminal control
4          ; sequence.
5          ;
6          ; Inputs:
7          ;   R1 = Line index number.
8          ;   SLCSBF = Control sequence in asciz form.
9          ;
10         CSFIN:  MOV      R2, -(SP)
11         ;
12         ; Say we are no longer processing a control sequence
13         ;
14         001356 010246          CLR      SLCSR          ; No longer accruing a control sequence
15         ;
16         ; Begin loop to search for the control sequence
17         ;
18         001364 012705 001456'  MOV      #CSTBL, R5      ; Get pointer to control sequence table
19         ;
20         ; Compare accrued control sequence with sequence stored in table
21         ;
22         001370 012702 000000G  1$:      MOV      #SLCSBF, R2      ; Point to accrued control sequence
23         001374 122225          2$:      CMPB      (R2)+, (R5)+      ; Compare the strings
24         001376 001004          BNE      3$          ; Br if mismatch
25         001400 105765 177777    TSTB      -1(R5)          ; Was that the last char of both strings?
26         001404 001373          BNE      2$          ; Br if not
27         001406 000415          BR       4$          ; We have a match
28         ;
29         ; Strings do not match -- Move on to next table entry
30         ;
31         001410 005305          3$:      DEC      R5          ; Point to last char compared in table
32         001412 105725          5$:      TSTB      (R5)+          ; Search for null at end of asciz string
33         001414 001376          BNE      5$
34         001416 062705 0000003    ADD      #3, R5          ; Bound up and skip over following word
35         001422 042705 0000001    BIC      #1, R5          ; Get to word boundary
36         001426 020527 002206'    CMP      R5, #CSEND      ; Checked all entries in table?
37         001432 103756          BLD      1$          ; Loop if not
38         ;
39         ; We could not find the sequence in our table
40         ;
41         001434 004737 011154'    CALL     RNGBEL          ; Ring bell
42         001440 000404          BR       9$
43         ;
44         ; We found control sequence in our table.
45         ; Call processing routine.
46         ;
47         001442 005205          4$:      INC      R5          ; Bound up to next word following string
48         001444 042705 0000001    BIC      #1, R5          ; Get to word boundary
49         001450 004735          CALL     @(R5)+          ; Call processing routine
50         ;
51         ; Finished
52         ;
53         001452 012602          9$:      MOV      (SP)+, R2
54         001454 000207          RETURN

```

CSFIN -- Process terminal control sequence

```

1      ;
2      ; Table of terminal control sequences and associated processing routines
3      ;
4      .MACRO ESCSEQ STRING,RTN
5      .ASCIZ /STRING/
6      .EVEN
7      .WORD RTN
8      .ENDM ESCSEQ
9
10     CSTBL:
11     ESCSEQ <[A],TCUP          ; Up arrow
12     ESCSEQ <[B],TCDOWN       ; Down arrow
13     ESCSEQ <[C],TCRITE       ; Right arrow
14     ESCSEQ <[D],TCLEFT       ; Left arrow
15     ESCSEQ <[OM],TCENTR      ; Enter (treat like carriage return)
16     ESCSEQ <[OP],TCPF1       ; PF1
17     ESCSEQ <[OQ],TCPF2       ; PF2
18     ESCSEQ <[OR],TCPF3       ; PF3
19     ESCSEQ <[OS],TCPF4       ; PF4
20     ESCSEQ <[O1],KEYCOM       ; Comma
21     ESCSEQ <[Om],KEYMIN      ; Minus sign
22     ESCSEQ <[On],KEYDOT      ; Period
23     ESCSEQ <[Op],KEY0        ; 0
24     ESCSEQ <[Oq],KEY1        ; 1
25     ESCSEQ <[Or],KEY2        ; 2
26     ESCSEQ <[Os],KEY3        ; 3
27     ESCSEQ <[Ot],KEY4        ; 4
28     ESCSEQ <[Ou],KEY5        ; 5
29     ESCSEQ <[Ov],KEY6        ; 6
30     ESCSEQ <[Ow],KEY7        ; 7
31     ESCSEQ <[Ox],KEY8        ; 8
32     ESCSEQ <[Oy],KEY9        ; 9
33     ESCSEQ <[1~],E1          ; E1 -- Find
34     ESCSEQ <[2~],E2          ; E2 -- Insert here
35     ESCSEQ <[3~],E3          ; E3 -- Remove
36     ESCSEQ <[4~],E4          ; E4 -- Select
37     ESCSEQ <[5~],E5          ; E5 -- Prev screen
38     ESCSEQ <[6~],E6          ; E6 -- Next screen
39     ESCSEQ <[17~],F6         ; F6
40     ESCSEQ <[18~],F7         ; F7
41     ESCSEQ <[19~],F8         ; F8
42     ESCSEQ <[20~],F9         ; F9
43     ESCSEQ <[21~],F10        ; F10
44     ESCSEQ <[23~],F11        ; F11 -- ESC
45     ESCSEQ <[24~],F12        ; F12 -- BS
46     ESCSEQ <[25~],F13        ; F13 -- Line feed
47     ESCSEQ <[26~],F14        ; F14
48     ESCSEQ <[28~],F15        ; F15 -- Help
49     ESCSEQ <[29~],F16        ; F16 -- Do
50     ESCSEQ <[31~],F17        ; F17
51     ESCSEQ <[32~],F18        ; F18
52     ESCSEQ <[33~],F19        ; F19
53     ESCSEQ <[34~],F20        ; F20
54     ESCSEQ <[OA],TCUP        ; Up arrow (Cursor key mode set)
55     ESCSEQ <[OB],TCDOWN      ; Down arrow (Cursor key mode set)
56     ESCSEQ <[OC],TCRITE      ; Right arrow (Cursor key mode set)
57     ESCSEQ <[OD],TCLEFT      ; Left arrow (Cursor key mode set)

```

58 002146	ESCSEQ	<A>, TCUP	; Up arrow	(VT52)
59 002152	ESCSEQ	, TCDOWN	; Down arrow	(VT52)
60 002156	ESCSEQ	<C>, TCRITE	; Right arrow	(VT52)
61 002162	ESCSEQ	<D>, TCLEFT	; Left arrow	(VT52)
62 002166	ESCSEQ	<P>, TCPF1	; PF1	(VT52 numeric)
63 002172	ESCSEQ	<Q>, TCPF2	; PF2	(VT52 numeric)
64 002176	ESCSEQ	<R>, TCPF3	; PF3	(VT52 numeric)
65 002202	ESCSEQ	<S>, TCPF4	; PF4	(VT52 numeric)
66 002206	CSEND:			

TCUP -- Up-arrow processing

```

1          .SBTTL  TCUP  -- Up-arrow processing
2          ;-----
3          ; Process the up-arrow key.
4          ;
5          ; Inputs:
6          ;   R1 = Job index number.
7          ;
8 002206 010546 TCUP:  MOV      R5, -(SP)
9          ;
10         ; See if this is a user-defined key
11         ;
12 002210         KEYCHK  KC$UP      ; See if this is a user-defined key
13 002216 103450      BCS      9$      ; Br if user-defined key
14         ;
15         ; If gold key was pressed recall through cycle
16         ;
17 002220 105737 000000G      TSTB     SLGOLD      ; Was gold key pressed?
18 002224 001426      BEQ      3$      ; Br if not
19 002226 105037 000000G      CLRB     SLGOLD      ; Clear gold key flag
20 002232 005737 000000G      TST      SLCYC1     ; Is a cycle defined?
21 002236 001421      BEQ      3$      ; Br if not
22 002240 013705 000000G      MOV      SLLPTR, R5   ; Point to last recalled line
23 002244 020537 000000G      CMP      R5, SLCYC2   ; Reached end of the cycle?
24 002250 001003      BNE      4$      ; Br if not
25 002252 013705 000000G      MOV      SLCYC1, R5   ; Loop back to start of cycle
26 002256 000402      BR        5$
27 002260 004737 007614' 4$:  CALL     SLMVDN     ; Move to next saved line
28 002264 010537 000000G 5$:  MOV      R5, SLLPTR   ; Remember last recalled line
29 002270 010537 000000G      MOV      R5, SLSPTR   ; Say this is last saved line
30 002274 004737 007704'      CALL     RSTLIN     ; Restore the line
31 002300 000417      BR        9$
32         ;
33         ; Get pointer to saved line
34         ;
35 002302 013705 000000G 3$:  MOV      SLLPTR, R5   ; Point to saved-line buffer
36         ;
37         ; If last recall was with down arrow, skip over a command
38         ;
39 002306 105737 000000G      TSTB     SLDOWN     ; Last operation down arrow?
40 002312 001404      BEQ      1$      ; Br if not
41 002314 105037 000000G      CLRB     SLDOWN     ; Reset direction indicator
42 002320 004737 007526'      CALL     SLMVUP     ; Move up 1 line
43         ;
44         ; Restore the saved line
45         ;
46 002324 004737 007704' 1$:  CALL     RSTLIN     ; Restore the line
47         ;
48         ; Update up-arrow pointer to point to next saved line
49         ;
50 002330 004737 007526'      CALL     SLMVUP     ; Move up to next line
51 002334 010537 000000G      MOV      R5, SLLPTR   ; Save pointer for next up-arrow
52         ;
53         ; Finished
54         ;
55 002340 012605 9$:  MOV      (SP)+, R5
56 002342 000207      RETURN

```

TCDOWN -- Down-arrow processing

```

1          .SBTTL  TCDOWN -- Down-arrow processing
2          ;-----
3          ; Process the down-arrow key -- Move forward to next saved command.
4          ;
5 002344    010446    TCDOWN: MOV      R4,-(SP)
6 002346    010546    MOV      R5,-(SP)
7          ;
8          ; See if this is a user-defined key
9          ;
10 002350          KEYCHK  KC$DWN          ;See if this is a user-defined key
11 002356    103454    BCS      9$          ;Br if user-defined key
12          ;
13          ; See if Gold key was pressed
14          ;
15 002360    105737    000000G    TSTB     SLGOLD          ;Was gold key pressed?
16 002364    001423    BEQ      3$          ;Br if not
17          ;
18          ; Gold-down-arrow -- Set recall cycle
19          ;
20 002366    105037    000000G    CLRB     SLGOLD          ;Reset Gold flag
21 002372    013705    000000G    MOV      SLLPTR,R5        ;Point to last recalled command
22 002376    105737    000000G    TSTB     SLDOWN          ;Was last direction down?
23 002402    001002    BNE      4$          ;Br if yes
24 002404    004737    007614'    CALL     SLMVDN          ;Move down a line
25 002410    010537    000000G    4$:  MOV      R5,SLCYC1      ;Set pointer to 1st command in cycle
26 002414    013737    000000G 000000G    MOV      SLSPTR,SLCYC2 ;Save pointer to last command in cycle
27 002422    010537    000000G    MOV      R5,SLSPTR        ;Say this is the last command entered
28 002426    010537    000000G    MOV      R5,SLLPTR        ;Remember last recalled line
29 002432    000426    BR       9$
30          ;
31          ; If last recalled command was done with up-arrow skip over that command
32          ;
33 002434    013705    000000G    3$:  MOV      SLLPTR,R5        ;Point to last saved command
34 002440    105737    000000G    TSTB     SLDOWN          ;Was last direction down?
35 002444    001005    BNE      1$          ;Br if yes
36 002446    112737    000001 000000G    MOVB     #1,SLDOWN      ;We are going down now
37 002454    004737    007614'    CALL     SLMVDN          ;Move down a command line
38          ;
39          ; See if there is a command to recall
40          ;
41 002460    020537    000000G    1$:  CMP      R5,SLSPTR        ;Any commands left to recall?
42 002464    001003    BNE      2$          ;Br if yes
43 002466    004737    011154'    CALL     RINGBEL        ;Ring the bell
44 002472    000406    BR       9$
45          ;
46          ; Advance down to the next saved command
47          ;
48 002474    004737    007614'    2$:  CALL     SLMVDN          ;Move down to next saved command
49          ;
50          ; Store new current line pointer
51          ;
52 002500    010537    000000G    MOV      R5,SLLPTR        ;Points to last line recalled
53          ;
54          ; Restore the saved line
55          ;
56 002504    004737    007704'    CALL     RSTLIN          ;Restore line pointed to by R5
57          ;

```

```
58          ; Finished
59          ;
60 002510    012605      9$:      MOV      (SP)+, R5
61 002512    012604              MOV      (SP)+, R4
62 002514    000207              RETURN
```

1			.SBTTL TCLEFT -- Left-arrow processing
2			-----
3			; Process left-arrow key.
4			;
5			; Inputs:
6			; R1 = Job index number.
7			;
8	002516	010446	TCLEFT: MOV R4, -(SP)
9			;
10			; See if this is a user-defined key
11			;
12	002520		KEYCHK KC\$LFT ; See if this is a user-defined key
13	002526	103422	BCS 9\$; Br if user-defined key
14			;
15			; If Gold key was pressed, move to left end of buffer
16			;
17	002530	105737 0000000	TSTB SLGOLD ; Was Gold key pressed?
18	002534	001407	BEG 2\$; Br if not
19	002536	012704 0000000	MOV #SLEBUF, R4 ; Position to left end of buffer
20	002542	004737 010642'	CALL CHRPOS ; Position cursor
21	002546	105037 0000000	CLRB SLGOLD ; Clear gold-key flag
22	002552	000410	BR 9\$
23			;
24			; Gold key was not pressed -- Move cursor left one character.
25			; Error if already at left margin.
26			;
27	002554	013704 0000000	2\$: MOV SLCX, R4 ; Get cursor position index
28	002560	020427 0000000	CMP R4, #SLEBUF ; Are we at left margin now?
29	002564	101403	BLOS 9\$; Br if yes -- Nothing to do
30			;
31			; Move cursor left 1 character
32			;
33	002566	005304	1\$: DEC R4 ; Move character pointer back 1 character
34	002570	004737 010642'	CALL CHRPOS ; Position cursor correctly
35			;
36			; Finished
37			;
38	002574	012604	9\$: MOV (SP)+, R4
39	002576	000207	RETURN

TCRITE -- Right-arrow processing

```

1          .SBTTL  TCRITE -- Right-arrow processing
2          ;-----
3          ; Process the right arrow key.
4          ;
5          ; Inputs:
6          ;   R1 = Job index.
7          ;
8 002600 010446 TCRITE: MOV      R4,-(SP)
9          ;
10         ; See if this is a user-defined key
11         ;
12 002602         KEYCHK  KC$RIT      ;See if this is a user-defined key
13 002610 103424     BCS      9$      ;Br if user-defined key
14         ;
15         ; If Gold key was pressed, move to right end of line.
16         ;
17 002612 105737 000000G     TSTB     SLGOLD      ;Was gold key pressed?
18 002616 001412     BEQ      1$      ;Br if not
19 002620 013704 000000G     MOV      SLCX,R4      ;Get current cursor pointer
20 002624 105724 3$: TSTB     (R4)+      ;Search for right end of line
21 002626 001376     BNE      3$      ;Loop till null hit
22 002630 005304     DEC      R4      ;Point to null at end
23 002632 004737 010642'     CALL     CHRPOS      ;Position cursor on screen
24 002636 105037 000000G     CLRB     SLGOLD      ;Clear gold-key flag
25 002642 000407     BR       9$
26         ;
27         ; Gold key was not pressed. Move cursor right 1 character.
28         ; Error if already at right end of line.
29         ;
30 002644 013704 000000G 1$: MOV      SLCX,R4      ;Get current cursor character pointer
31 002650 105714     TSTB     (R4)      ;Are we at right end of line now?
32 002652 001403     BEQ      9$      ;Br if yes
33         ;
34         ; Move cursor right 1 character.
35         ;
36 002654 005204 2$: INC      R4      ;Advance cursor 1 character
37 002656 004737 010642'     CALL     CHRPOS      ;Move cursor on screen
38         ;
39         ; Finished
40         ;
41 002662 012604 9$: MOV      (SP)+,R4
42 002664 000207     RETURN
43

```

```
1          .SBTTL  TCPF1  -- PF1 processing
2          ;-----
3          ; Process the PF1 (Gold) key.
4          ;
5          ; Inputs:
6          ;   R1 = Job index number.
7          ;
8 002666    TCPF1:
9          ;
10         ; See if this is a user-defined key
11         ;
12         KEYCHK  KC$PF1          ; See if this is a user-defined key
13 002674    103410      BCS      9$          ; Br if user-defined key
14         ;
15         ; See if PF1 has been pressed previously
16         ;
17 002676    105737    000000G      TSTB      SLGOLD          ; Has PF1 already been pressed?
18 002702    001403      BEQ      1$          ; Br if not
19 002704    105037    000000G      CLRB      SLGOLD          ; Clear gold-key flag
20 002710    000402      BR      9$
21         ;
22         ; Set flag saying Gold key pressed.
23         ;
24 002712    105237    000000G    1$:      INCB      SLGOLD          ; Remember "Gold" key was pressed
25         ;
26         ; Finished
27         ;
28 002716    000207    9$:      RETURN
```

```
1          .SBTTL  TCPF2  -- PF2 processing
2          ;-----
3          ; Process the PF2 key.
4          ;
5          ; Inputs
6          ; R1 = Job index number.
7          ;
8 002720    TCPF2:
9          ;
10         ; See if this is a user-defined key
11         ;
12 002720         KEYCHK  KC$PF2          ;See if this is a user-defined key
13 002726 103404      BCS      9$          ;Br if user-defined key
14         ;
15         ; PF2 is not defined if it is not a user-defined key.
16         ;
17 002730 004737 011154'      CALL  RNGBEL          ;Help support not implemented
18 002734 105037 000000G      CLR B   SLGOLD        ;Clear gold flag
19         ;
20         ; Finished
21         ;
22 002740 000207      9$:      RETURN
```

```
1          .SBTTL  TCPF3  -- PF3 processing
2          ;-----
3          ; Process PF3 key.
4          ;
5          ; Inputs:
6          ;   R1 = Job index number.
7          ;
8 002742    TCPF3:
9          ;
10         ; See if this is a user-defined key
11         ;
12 002742    KEYCHK  KC$PF3          ; See if this is a user-defined key
13 002750    BCS     9$              ; Br if user-defined key
14         ;
15         ; If this is a VT52, treat PF3 like PF4.
16         ;
17 002752    004737  011602'        CALL  CHK52          ; Is this a VT52 terminal?
18 002756    103405          BCS     TCPF4          ; Br if yes
19         ;
20         ; This is not a VT52
21         ;
22 002760    004737  011154'        CALL  RNGBEL          ; PF3 not valid for VT100
23 002764    105037  000000G        CLRB   SLGOLD        ; Clear gold-key flag
24 002770    000207          9$:    RETURN
```

1				.SBTTL TCPF4 -- PF4 processing
2				-----
3				; Process PF4 key.
4				;
5				; Inputs:
6				; R1 = Job index number.
7				;
8	002772	010246		TCPF4: MOV R2,-(SP)
9	002774	010346		MOV R3,-(SP)
10	002776	010446		MOV R4,-(SP)
11				;
12				; See if this is a user-defined key
13				;
14	003000			KEYCHK KC\$PF4 ;See if this is a user-defined key
15	003006	103430		BCS 9\$;Br if user-defined key
16				;
17				; See if Gold key was pressed
18				;
19	003010	013704	000000G	MOV SLCX,R4 ;Get current cursor position pointer
20	003014	105737	000000G	TSTB SLGOLD ;Was Gold key pressed?
21	003020	001013		BNE 1\$;Br if yes
22				;
23				; Gold key was not pressed.
24				; Delete from cursor to end of line.
25				;
26	003022	105714		TSTB (R4) ;Is cursor at right end of line now?
27	003024	001421		BEQ 9\$;Br if yes
28				;
29				; Move deleted characters to holding buffer.
30				;
31	003026	010402		2\$: MOV R4,R2 ;Get cursor pointer
32	003030	012703	000000G	MOV #SLDBUF,R3 ;Point to delete buffer
33	003034	112223		3\$: MOVB (R2)+,(R3)+ ;Move deleted chars to holding buffer
34	003036	001376		BNE 3\$;Loop till null moved
35				;
36				; Truncate line at cursor position
37				;
38	003040	105014		CLRB (R4) ;Truncate line
39				;
40				; Display truncated line
41				;
42	003042	004737	010336'	CALL PAINT ;Redisplay line from cursor to right end
43	003046	000410		BR 9\$
44				;
45				; Gold (PF4) key was pressed. Replace text from holding buffer.
46				; Insert text to right of cursor position.
47				;
48	003050	012702	000000G	1\$: MOV #SLDBUF,R2 ;Get pointer to holding buffer
49	003054	004737	010140'	CALL INSERT ;Insert text
50	003060	004737	010642'	CALL CHRPOS ;Position cursor
51	003064	105037	000000G	CLRB SLGOLD ;Clear gold-key flag
52				;
53				; Finished
54				;
55	003070	012604		9\$: MOV (SP)+,R4
56	003072	012603		MOV (SP)+,R3
57	003074	012602		MOV (SP)+,R2

TSSLE -- TSX-Plus Single Line E MACRO V05.05 Wednesday 18-Jan-89 14:16 Page 21-1
TCPF4 -- PF4 processing

58 003076 000207

RETURN

```
1          .SBTTL  TCENTR -- Enter key processing
2          ;-----
3          ; Treat Enter key like carriage-return line-feed.
4          ;
5 003100    TCENTR:
6          ;
7          ; See if this is a user-defined key
8          ;
9 003100          KEYCHK  KC$ENT          ; See if this is a user-defined key
10 003106 103406    BCS      9$          ; Br if user-defined key
11          ;
12          ; This is not a user-defined key
13          ;
14 003110 105037 000000G    CLRB      SLGOLD          ; Reset gold key
15 003114 105237 000000G    INCB      SLCR           ; Say we have received carriage-return
16 003120 004737 005622'    CALL      ICPCR          ; Process carriage-return, line-feed
17 003124 000207          9$: RETURN
18
19          .SBTTL  TCINVL -- Invalid function key
20          ;-----
21          ; An invalid function key was pressed.
22          ;
23 003126    TCINVL:
24          ;
25          ; Ring terminal bell
26          ;
27 003126 004737 011154'    CALL      RRGBEL          ; Ring the bell
28          ;
29          ; Clear gold-key flag
30          ;
31 003132 105037 000000G    CLRB      SLGOLD          ; Clear gold-key flag
32          ;
33          ; Finished
34          ;
35 003136 000207          RETURN
```

KEYO -- Key "O" processing

```

1          .SBTTL  KEYO  -- Key "O" processing
2          ;-----
3          ; Key O -- Bline / Open line
4          ;
5 003140    010246
6 003142    010346
7 003144    010446
8          ;
9          ; See if this is a user-defined key
10         ;
11         KEYCHK  KC$KPO          ; See if this is a user-defined key
12 003154    103426              BCS  9$          ; Br if user-defined key
13         ;
14         ; See if the gold key was pressed
15         ;
16 003156    105737    000000G    TSTB  SLGOLD          ; Was the gold key pressed?
17 003162    001005              BNE  1$          ; Br if yes
18         ;
19         ; Gold key was not pressed.
20         ; Move cursor to front of line.
21         ;
22 003164    012704    000000G    MOV  #SLEBUF,R4      ; Set cursor to front of line
23 003170    004737    010642'    CALL  CHRPOS        ; Position the cursor
24 003174    000416              BR   9$
25         ;
26         ; Gold key was pressed.
27         ; Delete from cursor to end of line.
28         ;
29 003176    105037    000000G    1$:  CLRB  SLGOLD      ; Clear gold-key flag
30 003202    013704    000000G    MOV  SLCX,R4        ; Get cursor position index
31 003206    105714              TSTB  (R4)          ; Are we already at the end of the line?
32 003210    001410              BEQ  9$          ; Br if yes
33         ;
34         ; Move deleted characters to holding buffer
35         ;
36 003212    010402              MOV  R4,R2          ; Get cursor pointer
37 003214    012703    000000G    MOV  #SLDBUF,R3      ; Point to holding buffer
38 003220    112223    3$:  MOVB  (R2)+,(R3)+        ; Move chars to holding buffer
39 003222    001376              BNE  3$          ; Loop till null moved
40         ;
41         ; Truncate the line
42         ;
43 003224    105014              CLRB  (R4)          ; Truncate the line at the cursor position
44         ;
45         ; Display the truncated line
46         ;
47 003226    004737    010336'    CALL  PAINT        ; Redisplay the line
48         ;
49         ; Finished
50         ;
51 003232    012604    9$:  MOV  (SP)+,R4
52 003234    012603              MOV  (SP)+,R3
53 003236    012602              MOV  (SP)+,R2
54 003240    000207              RETURN

```

KEY1 -- Key "1" processing

```

1          .SBTTL  KEY1  -- Key "1" processing
2          -----
3          ; Key 1 -- Word / Change case
4          ;
5 003242 010446 KEY1:  MOV      R4, -(SP)
6          ;
7          ; See if this is a user-defined key
8          ;
9 003244          KEYCHK  KC$KP1          ; See if this is a user-defined key
10 003252 103526      BCS      20$          ; Br if user-defined key
11          ;
12          ; See if the gold key was pressed
13          ;
14 003254 105737 000000G TSTB      SLGOLD          ; Was the gold key pressed?
15 003260 001053      BNE      1$          ; Br if yes
16          ;
17          ; Gold key was not pressed.
18          ; Move forward/backward one word.
19          ;
20 003262 013704 000000G MOV      SLCX, R4          ; Get current cursor index
21 003266 105737 000000G TSTB      SLBACK          ; Moving forward or backward?
22 003272 001021      BNE      16$          ; Br if backward
23          ;
24          ; Move forward one word.
25          ; See if cursor is pointing to a delimiter now.
26          ;
27 003274 112400      MOVVB    (R4)+, R0          ; Get char under cursor
28 003276 001514      BEQ      20$          ; Br if at end of line now
29 003300 004737 011010' CALL      CHKDLM          ; Is current char a delimiter?
30 003304 103405      BCS      4$          ; Br if yes
31          ;
32          ; Skip characters in word under cursor
33          ;
34 003306 112400      MOVVB    (R4)+, R0          ; Get next char from word
35 003310 001410      BEQ      6$          ; Br if hit end of line
36 003312 004737 011010' CALL      CHKDLM          ; Is this char a delimiter?
37 003316 103373      BCC      5$          ; Br if not
38          ;
39          ; Skip delimiters that follow the word
40          ;
41 003320 112400      MOVVB    (R4)+, R0          ; Get next char
42 003322 001403      BEQ      6$          ; Br if hit end of line
43 003324 004737 011010' CALL      CHKDLM          ; Is this a delimiter?
44 003330 103773      BCS      4$          ; Loop to skip all delimiters following word
45 003332 005304      6$: DEC      R4          ; Point to 1st char of new word
46 003334 000422      BR       10$
47          ;
48          ; Move backward one word
49          ;
50 003336 020427 000000G 16$: CMP      R4, #SLEBUF          ; Are we already at left end of line?
51 003342 101472      BLOS     20$          ; Br if yes
52          ;
53          ; Skip over delimiters to left of cursor.
54          ;
55 003344 020427 000000G 7$: CMP      R4, #SLEBUF          ; Are we at left end of line now?
56 003350 101414      BLOS     10$          ; Br if yes
57 003352 114400      MOVVB    -(R4), R0          ; Get next char to left

```

KEY1 -- Key "1" processing

```

58 003354 004737 011010'      CALL    CHKDLM      ;Is this a delimiter?
59 003360 103771              BCS      7$          ;Loop to skip over delimiters
60                               ;
61                               ; Skip over characters in the word
62                               ;
63 003362 020427 000000G      8$:    CMP      R4,#SLEBUF    ;Are we at left end of line yet?
64 003366 101405              BLOS     10$          ;Br if yes
65 003370 114400              MOV      -(R4),R0        ;Get next char to left
66 003372 004737 011010'      CALL    CHKDLM      ;Is this a delimiter?
67 003376 103371              BCC      8$          ;Loop if not
68                               ;
69                               ; Point to 1st char of the word
70                               ;
71 003400 005204              INC      R4              ;Point to 1st char of the word
72                               ;
73                               ; Position cursor at start of word
74                               ;
75 003402 004737 010642'      10$:    CALL    CHRPOS      ;Position cursor at 1st char of word
76 003406 000450              BR       20$
77                               ;
78                               ; Gold key 1 -- Change case of character under the cursor
79                               ;
80 003410 105037 000000G      1$:    CLRB     SLGOLD      ;Clear the gold-key flag
81 003414 013704 000000G      MOV      SLCX,R4        ;Get current cursor index
82 003420 111400              MOV      (R4),R0        ;Get char under the cursor
83 003422 001440              BEQ      19$          ;Br if at right end of line
84 003424 020027 000141      CMP      R0,#141        ;Is this a lower-case letter?
85 003430 103406              BLO      11$          ;Br if not
86 003432 020027 000172      CMP      R0,#172        ;
87 003436 101016              BHI      15$          ;Br if not
88 003440 162700 000040      SUB      #40,R0        ;Convert lower-case to upper-case
89 003444 000410              BR       14$
90 003446 020027 000101      11$:    CMP      R0,#'A    ;Is this a upper-case letter?
91 003452 103410              BLO      15$          ;Br if not
92 003454 020027 000132      CMP      R0,#'Z
93 003460 101005              BHI      15$          ;Br if not
94 003462 062700 000040      ADD      #40,R0        ;Convert upper-case to lower-case
95 003466 110014              14$:    MOV      R0,(R4)    ;Store converted character
96 003470 004737 010336'      CALL    PAINT        ;Redisplay the line
97                               ;
98                               ; Advance cursor to next character
99                               ;
100 003474 105737 000000G     15$:    TSTB     SLBACK      ;Moving forward or backward?
101 003500 001002              BNE      12$          ;Br if backward
102 003502 005204              INC      R4              ;Advance cursor pointer
103 003504 000404              BR       13$
104 003506 020427 000000G     12$:    CMP      R4,#SLEBUF    ;Are we at left end of line now?
105 003512 101401              BLOS     13$          ;Br if yes
106 003514 005304              DEC      R4              ;Backup the cursor
107 003516 004737 010642'     13$:    CALL    CHRPOS      ;Position cursor to next char
108 003522 000402              BR       20$
109                               ;
110                               ; Error -- ring bell
111                               ;
112 003524 004737 011154'     19$:    CALL    RNGBEL      ;Ring the bell
113                               ;
114                               ; Finished

```

```
115  
116 003530 012604      ;  
117 003532 000207      20$:  MOV      (SP)+, R4  
                          RETURN
```

KEY2 -- Key "2" processing

```

1          .SBTTL  KEY2  -- Key "2" processing
2          ;-----
3          ; Key "2" -- EOL / Del EOL
4          ;
5 003534 010446 KEY2:  MOV      R4,-(SP)
6          ;
7          ; See if this is a user-defined key
8          ;
9 003536          KEYCHK  KC$KP2          ; See if this is a user-defined key
10 003544 103427      BCS      20$          ; Br if user-defined key
11          ;
12          ; See if gold key was pressed
13          ;
14 003546 105737 000000G      TSTB      SLGOLD          ; Was the gold key pressed?
15 003552 001020      BNE      1$          ; Br if yes
16          ;
17          ; Gold key was not pressed.
18          ; Check which direction to move.
19          ;
20 003554 105737 000000G      TSTB      SLBACK          ; Move to left or right end of line?
21 003560 001010      BNE      2$          ; Br if moving to left end
22          ;
23          ; Move to right end of line.
24          ;
25 003562 013704 000000G      MOV      SLCX,R4          ; Get current cursor pointer
26 003566 105724      3$:      TSTB      (R4)+          ; Search for null at the end of the line
27 003570 001376      BNE      3$
28 003572 005304      DEC      R4          ; Point to the null
29 003574 004737 010642'      CALL      CHRPOS          ; Position cursor to end of line
30 003600 000411      BR      20$
31          ;
32          ; Move to left end of line.
33          ;
34 003602 012704 000000G      2$:      MOV      #SLEBUF,R4          ; Point to left end of line
35 003606 004737 010642'      CALL      CHRPOS          ; Position cursor there
36 003612 000404      BR      20$
37          ;
38          ; Gold key 2 -- Delete to end of line
39          ;
40 003614 105037 000000G      1$:      CLRB      SLGOLD          ; Clear gold-key flag
41 003620 004737 002772'      CALL      TCPF4          ; Process exactly like PF4 key
42          ;
43          ; Finished
44          ;
45 003624 012604      20$:      MOV      (SP)+,R4
46 003626 000207      RETURN

```

KEY3 -- Key "3" processing

```

1          .SBTTL  KEY3  -- Key "3" processing
2          ;-----
3          ; Key "3" -- VT100: Advance by character, VT52: invalid
4          ;
5          ; Inputs:
6          ;   R1 = Job index number.
7          ;
8 003630    010446  KEY3:  MOV      R4, -(SP)
9          ;
10         ; See if this is a user-defined key
11         ;
12 003632         KEYCHK  KC$KP3          ; See if this is a user-defined key
13 003640    103432  BCS      9$          ; Br if user-defined key
14         ;
15         ; Don't allow gold key with key 3.
16         ;
17 003642    105737    000000G  TSTB     SLGOLD          ; Was gold key pressed?
18 003646    001023  BNE      8$          ; Br if yes -- invalid
19         ;
20         ; Key 3 is invalid for VT52
21         ;
22 003650    004737    011602'  CALL     CHK52          ; Is this a VT52?
23 003654    103420  BCS      8$          ; Br if it is
24         ;
25         ; Advance/Backup 1 character
26         ;
27 003656    013704    000000G  MOV      SLCX, R4          ; Get current cursor pointer
28 003662    105737    000000G  TSTB     SLBACK          ; Advance or backup?
29 003666    001004  BNE      1$          ; Br if backup
30         ;
31         ; Advance one character
32         ;
33 003670    105714  TSTB     (R4)          ; Are we at right end of line now?
34 003672    001415  BEQ      9$          ; Br if yes
35 003674    005204  INC      R4          ; Advance cursor
36 003676    000404  BR       2$          ;
37         ;
38         ; Backup cursor one character
39         ;
40 003700    020427    000000G  1$:  CMP      R4, #SLEBUF      ; Are we at left end of line now?
41 003704    101410  BLOS     9$          ; Br if yes
42 003706    005304  DEC      R4          ; Backup the cursor
43 003710    004737    010642'  2$:  CALL     CHRPOS          ; Display cursor at correct position
44 003714    000404  BR       9$          ;
45         ;
46         ; Error -- Ring the bell
47         ;
48 003716    004737    011154'  8$:  CALL     RNGBEL          ; Ring the bell
49 003722    105037    000000G  CLRB     SLGOLD          ; Clear the gold key flag
50         ;
51         ; Finished
52         ;
53 003726    012604  9$:  MOV      (SP)+, R4
54 003730    000207  RETURN

```

KEY4 -- Key "4" processing

```

1          .SBTTL  KEY4  -- Key "4" processing
2          ;-----
3          ; Key "4" -- Set forward direction / Move to right end of line.
4          ;
5 003732   KEY4:
6          ;
7          ; See if this is a user-defined key
8          ;
9 003732   KEYCHK  KC$KP4          ;See if this is a user-defined key
10 003740 103410   BCS          9$          ;Br if user-defined key
11          ;
12          ; See if the gold key was pressed
13          ;
14 003742 105737 000000G   TSTB  SLGOLD          ;Was gold key pressed?
15 003746 001003   BNE  1$          ;Br if yes
16          ;
17          ; Gold key was not pressed -- Set forward direction
18          ;
19 003750 105037 000000G   CLRB  SLBACK          ;Set forward direction
20 003754 000402   BR  9$
21          ;
22          ; Gold key was pressed -- Move to the right end of the line
23          ;
24 003756 004737 002600'  1$:  CALL  TCRITE          ;Process just like right-arrow key
25          ;
26          ; Finished
27          ;
28 003762 000207  9$:  RETURN

```

KEY5 -- Key "5" processing

```

1          .SBTTL  KEY5  -- Key "5" processing
2          ;-----
3          ; Key "5" -- Set backward direction / Move to left end of line
4          ;
5 003764    KEY5:
6          ;
7          ; See if this is a user-defined key
8          ;
9 003764    KEYCHK  KC$KP5          ;See if this is a user-defined key
10 003772 103411  BCS      9$          ;Br if user-defined key
11          ;
12          ; See if the gold key was pressed
13          ;
14 003774 105737 000000G    TSTB    SLGOLD          ;Was gold key pressed?
15 004000 001004    BNE      1$          ;Br if yes
16          ;
17          ; Gold key not pressed -- Set backward direction.
18          ;
19 004002 112737 000001 000000G    MOVB    #1,SLBACK      ;Set backward direction
20 004010 000402    BR      9$
21          ;
22          ; Gold key was pressed -- Move to left end of line
23          ;
24 004012 004737 002516'    1$:    CALL    TCLEFT          ;Treat just like left-arrow key
25          ;
26          ; Finished
27          ;
28 004016 000207    9$:    RETURN

```

```

1
2
3
4
5
6
7
8 004020
9
10
11
12 004020
13 004026 103411
14
15
16
17 004030 004737 011602'
18 004034 103002
19
20
21
22 004036 000137 004154'
23
24
25
26 004042 004737 011154'
27 004046 105037 0000000
28 004052 000207

      .SBTTL KEY6 -- Key "6" processing
      -----
      ; Key "6" -- VT100: invalid, VT52: [un] delete character
      ;
      ; Inputs:
      ; R1 = Job index number.
      ;
      KEY6:
      ;
      ; See if this is a user-defined key
      ;
      KEYCHK KC$KP6 ;See if this is a user-defined key
      BCS 9$ ;Br if user-defined key
      ;
      ; Determine if this is a VT52 or VT100
      ;
      CALL CHK52 ;Is this a VT52?
      BCC 1$ ;Br if not
      ;
      ; VT52 -- Treat key 6 like VT100 comma key
      ;
      JMP KEYCOM ;Treat like comma key
      ;
      ; VT100 -- invalid
      ;
      1$: CALL RNGBEL ;Invalid function
      CLRB SLGOLD ;Reset gold key flag
      9$: RETURN
  
```

```
1          .SBTTL  KEY7  -- Key "7" processing
2          ;-----
3          ; Key "7" -- No function
4          ;
5          ; Inputs:
6          ; R1 = Job index number
7          ;
8 004054    KEY7:
9          ;
10         ; See if this is a user-defined key
11         ;
12 004054    KEYCHK  KC$KP7          ;See if this is a user-defined key
13 004062 103404    BCS  9$          ;Br if user-defined key
14         ;
15         ; Not user-defined key
16         ;
17 004064 004737 011154'    CALL  RNGBEL      ;Ring the bell
18 004070 105037 000000G    CLRB  SLGOLD     ;Clear gold-key flag
19         ;
20         ; Finished
21         ;
22 004074 000207    9$:  RETURN
```

```
1          .SBTTL  KEY8  -- Key "8" processing
2          ;-----
3          ; Key "8" -- No function
4          ;
5          ; Inputs:
6          ; R1 = Job index number
7          ;
8 004076    KEY8:
9          ;
10         ; See if this is a user-defined key
11         ;
12 004076    KEYCHK  KC$KP8          ;See if this is a user-defined key
13 004104    BCS     9$              ;Br if user-defined key
14         ;
15         ; This is not a user-defined key
16         ;
17 004106    004737  011154'        CALL  RRGBEL          ;Ring the bell
18 004112    105037  000000G        CLRB   SLGOLD         ;Clear gold-key flag
19         ;
20         ; Finished
21         ;
22 004116    000207          9$:    RETURN
```

```

1
2
3
4
5
6
7
8 004120
9
10
11
12 004120
13 004126 103411
14
15
16
17 004130 004737 011602'
18 004134 103002
19
20
21
22 004136 000137 004262'
23
24
25
26 004142 004737 011154'
27 004146 105037 000000G
28 004152 000207

.SBTTL KEY9 -- Key "9" processing
-----
; Key "9" -- VT100: invalid, VT52: [un] delete word
;
; Inputs:
; R1 = Job index number.
;
KEY9:
; See if this is a user-defined key
;
; KEYCHK KC$KP9 ;See if this is a user-defined key
; BCS 9$ ;Br if user-defined key
;
; This is not a user-defined key
;
; CALL CHK52 ;Is this a VT52 terminal?
; BCC 1$ ;Br if not
;
; VT52 -- Treat like VT100 minus key
;
; JMP KEYMIN ;Treat like minus key
;
; VT100 -- Invalid
;
1$: CALL RINGBEL ;Ring the bell
; CLRB SLGOLD ;Clear gold-key flag
9$: RETURN

```

1			.SBTTL KEYCOM -- Key ", " processing
2			-----
3			; Comma key -- [un] delete character
4			;
5	004154	010246	KEYCOM: MOV R2,-(SP)
6	004156	010446	MOV R4,-(SP)
7			;
8			; See if this is a user-defined key
9			;
10	004160		KEYCHK KC\$COM ; See if this is a user-defined key
11	004166	103432	BCS 9\$; Br if user-defined key
12			;
13			; See if Gold key (PF1) was pressed.
14			;
15	004170	105737 000000G	TSTB SLGOLD ; Was Gold key pressed?
16	004174	001414	BEG 1\$; Br if not
17			;
18			; Gold key was pressed -- put back previously deleted character
19			;
20	004176	012702 000000G	MOV #SLCBUF,R2 ; Point to deleted-character buffer
21	004202	004737 010140'	CALL INSERT ; Insert the character
22	004206	013704 000000G	MOV SLCX,R4 ; Get current cursor pointer
23	004212	005304	DEC R4 ; Point to character we inserted
24	004214	004737 010642'	CALL CHRPOS ; Position cursor to inserted character
25	004220	105037 000000G	CLRB SLGOLD ; Clear Gold-key flag
26	004224	000413	BR 9\$
27			;
28			; Gold key not pressed -- Delete character under the cursor
29			;
30	004226	013704 000000G	1\$: MOV SLCX,R4 ; Get current cursor pointer
31	004232	105714	TSTB (R4) ; Is cursor at right end of line?
32	004234	001407	BEG 9\$; Br if yes
33			;
34			; Save character being deleted and then delete it
35			;
36	004236	111437 000000G	2\$: MOVB (R4),SLCBUF ; Save character being deleted
37	004242	005204	INC R4 ; Move cursor right 1 character
38	004244	004737 010642'	CALL CHRPOS
39	004250	004737 007454'	CALL DELLFT ; Now delete character to left of cursor
40			;
41			; Finished
42			;
43	004254	012604	9\$: MOV (SP)+,R4
44	004256	012602	MOV (SP)+,R2
45	004260	000207	RETURN

```

1          .SBTTL  KEYMIN -- Key "-" processing
2          ;-----
3          ; Minus key -- [un] delete word
4          ;
5 004262    010246    KEYMIN: MOV      R2,-(SP)
6 004264    010346          MOV      R3,-(SP)
7 004266    010446          MOV      R4,-(SP)
8 004270    010546          MOV      R5,-(SP)
9          ;
10         ; See if this is a user-defined key
11         ;
12 004272          KEYCHK  KC$MIN          ; See if this is a user-defined key
13 004300    103463          BCS      20$          ; Br if user-defined key
14         ;
15         ; See if the gold key was pressed
16         ;
17 004302    013704    000000G          MOV      SLGX,R4          ; Get current cursor index
18 004306    105737    000000G          TSTB     SLGOLD          ; Was the gold key pressed?
19 004312    001041          BNE      1$          ; Br if yes
20         ;
21         ; Gold key was not pressed -- Delete the word under the cursor
22         ;
23 004314    010405          MOV      R4,R5          ; Remember starting position of deletion
24 004316    112400          MOVB     (R4)+,R0        ; Get current char
25 004320    001453          BEQ      20$          ; Br if at right end of line now
26 004322    004737    011010'          CALL     CHKDLM        ; Is this character a delimiter?
27 004326    103405          BCS      3$          ; Br if yes
28 004330    112400    2$: MOVB     (R4)+,R0        ; Get next character
29 004332    001410          BEQ      4$          ; Br if hit right end of line
30 004334    004737    011010'          CALL     CHKDLM        ; Is this character a delimiter?
31 004340    103373          BCC      2$          ; Br if not
32         ;
33         ; Delete delimiters that follow the word
34         ;
35 004342    112400    3$: MOVB     (R4)+,R0        ; Get next character
36 004344    001403          BEQ      4$          ; Br if hit right end of line
37 004346    004737    011010'          CALL     CHKDLM        ; Is this a delimiter?
38 004352    103773          BCS      3$          ; Loop if yes
39 004354    005304    4$: DEC      R4          ; Point to 1st char of following word
40         ;
41         ; We have identified the characters to delete.
42         ; Move the characters to a holding buffer.
43         ;
44 004356    010503          MOV      R5,R3          ; Get index to 1st char to delete
45 004360    012702    000000G          MOV      #SLDBUF,R2        ; Point to holding buffer
46 004364    112322    5$: MOVB     (R3)+,(R2)+      ; Move char to holding buffer?
47 004366    020304          CMP      R3,R4          ; Moved all we need to delete?
48 004370    103775          BLO      5$          ; Loop if not
49 004372    105012          CLRB     (R2)          ; Put null at end of deleted char string
50         ;
51         ; Delete the characters from the buffer
52         ;
53 004374    010503          MOV      R5,R3          ; Get pointer to 1st char to delete
54 004376    112423    6$: MOVB     (R4)+,(R3)+      ; Move remainder of line left
55 004400    001376          BNE      6$          ; Loop till null at end moved
56         ;
57         ; Redisplay the line

```

```

58
59 004402 010504          ;      MOV      R5,R4          ;Get new position of cursor
60 004404 004737 010336'  ;      CALL     PAINT          ;Redisplay the line
61
62          ;      Reposition cursor
63
64 004410 004737 010642'  ;      CALL     CHRPOS          ;Resposition cursor
65 004414 000415          ;      BR       20$
66
67          ;      Gold minus key -- Replace deleted word
68
69 004416 105037 000000G 1$:      CLRB      SLGOLD          ;Reset gold-key flag
70
71          ;      Insert the deleted word
72
73 004422 013704 000000G          ;      MOV      SLCX,R4          ;Get current cursor index
74 004426 012702 000000G          ;      MOV      #SLDBUF,R2       ;Point to buffer with deleted text
75 004432 004737 010140'          ;      CALL     INSERT          ;Insert the text
76 004436 004737 010642'          ;      CALL     CHRPOS          ;Position cursor to front of replaced text
77 004442 000402          ;      BR       20$
78
79          ;      Error -- Ring bell
80
81 004444 004737 011154' 19$:      CALL     RNGBEL          ;Ring bell
82
83          ;      Finished
84
85 004450 012605 20$:      MOV      (SP)+,R5
86 004452 012604          ;      MOV      (SP)+,R4
87 004454 012603          ;      MOV      (SP)+,R3
88 004456 012602          ;      MOV      (SP)+,R2
89 004460 000207          ;      RETURN

```

KBS -- Save a command

```

1          .SBTTL  KBS      -- Save a command
2          ;-----
3          ; This routine is called when PF1 S is typed.  It saves a command.
4          ;
5 004462    010446    KBS:    MOV      R4,-(SP)
6 004464    010546          MOV      R5,-(SP)
7          ;
8          ; Save line
9          ;
10 004466    105037    000000G      CLRB    SLGOLD          ;Clear gold-key flag
11 004472    012704    000000G      MOV     #SLEBUF,R4      ;Point to current line buffer
12 004476    012705    000000G      MOV     #SLSBUF,R5      ;Point to save buffer
13 004502    112425          2$:     MOVB   (R4)+,(R5)+      ;Save the line
14 004504    001376          BNE     2$          ;Loop if more to save
15          ;
16          ; Finished
17          ;
18 004506    012605          MOV     (SP)+,R5
19 004510    012604          MOV     (SP)+,R4
20 004512    000207          RETURN

```

KBX -- Recall the saved command

```
1
2
3
4
5 004514 010546
6
7
8
9 004516 012705 000000G
10 004522 004737 007704'
11
12
13
14 004526 012605
15 004530 000207
```

```

.SBTTL KBX -- Recall the saved command
-----
; This routine is called when PF1 X is typed. It recalls the saved command.
;
KBX:  MOV     R5, -(SP)
;
; Insert saved line
;
      MOV     #SLSBUF, R5      ;Point to buffer with saved line
      CALL    RSTLIN          ;Restore the line
;
; Finished
;
      MOV     (SP)+, R5
      RETURN
```

E1 -- E1 processing

```
1          .SBTTL  E1      -- E1 processing
2          ;-----
3          ; Process E1 key.
4          ;
5          ; Inputs:
6          ; R1 = Job index number.
7          ;
8 004532    E1:
9          ;
10         ; See if this is a user-defined key
11         ;
12 004532    KEYCHK  KC$E1          ;See if this is a user-defined key
13 004540    BCS     9$            ;Br if user-defined key
14         ;
15         ; This is not a user-defined key.
16         ; Invalid key.
17         ;
18 004542    004737  011154'      CALL  RNGBEL          ;Ring the bell
19 004546    105037  000000G      CLR   SLGOLD         ;Clear gold-key flag
20         ;
21         ; Finished
22         ;
23 004552    000207      9$:      RETURN
```

```
1                                     .SBTTL  E2      -- E2 processing
2                                     -----
3                                     ; Process E2 key.
4                                     ;
5                                     ; Inputs:
6                                     ;   R1 = Job index number.
7                                     ;
8 004554                               E2:
9                                     ;
10                                    ; See if this is a user-defined key
11                                    ;
12 004554                               KEYCHK  KC$E2          ;See if this is a user-defined key
13 004562 103404                       BCS      9$            ;Br if user-defined key
14                                    ;
15                                    ; This is not a user-defined key.
16                                    ; Invalid key.
17                                    ;
18 004564 004737 011154'               CALL    RNGBEL        ;Ring the bell
19 004570 105037 000000G               CLR     SLGOLD        ;Clear gold-key flag
20                                    ;
21                                    ; Finished
22                                    ;
23 004574 000207                       9$:    RETURN
```

```
1          .SBTTL  E3      -- E3 processing
2          ;-----
3          ; Process E3 key.
4          ;
5          ; Inputs:
6          ; R1 = Job index number.
7          ;
8 004576    E3:
9          ;
10         ; See if this is a user-defined key
11         ;
12 004576    KEYCHK  KC$E3          ;See if this is a user-defined key
13 004604    BCS     9$            ;Br if user-defined key
14         ;
15         ; This is not a user-defined key.
16         ; Invalid key.
17         ;
18 004606    004737  011154'      CALL  RNGBEL          ;Ring the bell
19 004612    105037  000000G      CLR   SLGOLD         ;Clear gold-key flag
20         ;
21         ; Finished
22         ;
23 004616    000207      9$:      RETURN
```

```
1          .SBTTL  E4      -- E4 processing
2          ; -----
3          ; Process E4 key.
4          ;
5          ; Inputs:
6          ;   R1 = Job index number.
7          ;
8 004620    E4:
9          ;
10         ; See if this is a user-defined key
11         ;
12 004620          KEYCHK  KC$E4          ; See if this is a user-defined key
13 004626 103404      BCS    9$          ; Br if user-defined key
14         ;
15         ; This is not a user-defined key.
16         ; Invalid key.
17         ;
18 004630 004737 011154'      CALL  RNOBEL      ; Ring the bell
19 004634 105037 000000G      CLRB   SLGOLD     ; Clear gold-key flag
20         ;
21         ; Finished
22         ;
23 004640 000207      9$:    RETURN
```

```
1          .SBTTL  E5      -- E5 processing
2          ;-----
3          ; Process E5 key.
4          ;
5          ; Inputs:
6          ;   R1 = Job index number.
7          ;
8 004642    E5:
9          ;
10         ; See if this is a user-defined key
11         ;
12 004642    KEYCHK  KC$E5          ;See if this is a user-defined key
13 004650    BCS     9$            ;Br if user-defined key
14         ;
15         ; This is not a user-defined key.
16         ; Invalid key.
17         ;
18 004652    004737  011154'      CALL  RNGBEL          ;Ring the bell
19 004656    105037  000000G      CLR   SLGOLD         ;Clear gold-key flag
20         ;
21         ; Finished
22         ;
23 004662    000207      9$:      RETURN
```

```
1          .SBTTL  E6      -- E6 processing
2          ;-----
3          ; Process E6 key.
4          ;
5          ; Inputs:
6          ;   R1 = Job index number.
7          ;
8 004664    E6:
9          ;
10         ; See if this is a user-defined key
11         ;
12 004664          KEYCHK  KC$E6          ;See if this is a user-defined key
13 004672 103404    BCS     9$           ;Br if user-defined key
14         ;
15         ; This is not a user-defined key.
16         ; Invalid key.
17         ;
18 004674 004737 011154'    CALL  RNGBEL          ;Ring the bell
19 004700 105037 000000G    CLR    SLGOLD         ;Clear gold-key flag
20         ;
21         ; Finished
22         ;
23 004704 000207    9$:    RETURN
```

```
1          .SBTTL  F6      -- F6 processing
2          ;-----
3          ; Process F6 key.
4          ;
5          ; Inputs:
6          ; R1 = Job index number.
7          ;
8 004706    F6:
9          ;
10         ; See if this is a user-defined key
11         ;
12 004706    KEYCHK  KC$F6          ;See if this is a user-defined key
13 004714    BCS     9$            ;Br if user-defined key
14         ;
15         ; This is not a user-defined key.
16         ; Invalid key.
17         ;
18 004716    004737  011154'      CALL  RNGBEL          ;Ring the bell
19 004722    105037  000000G      CLR   SLGOLD         ;Clear gold-key flag
20         ;
21         ; Finished
22         ;
23 004726    000207      9$:      RETURN
```

```
1          .SBTTL  F7      -- F7 processing
2          ;-----
3          ; Process F7 key.
4          ;
5          ; Inputs:
6          ;   R1 = Job index number.
7          ;
8 004730    F7:
9          ;
10         ; See if this is a user-defined key
11         ;
12 004730         KEYCHK  KC$F7          ;See if this is a user-defined key
13 004736 103404     BCS    9$          ;Br if user-defined key
14         ;
15         ; This is not a user-defined key.
16         ; Invalid key.
17         ;
18 004740 004737 011154'     CALL  RNGBEL          ;Ring the bell
19 004744 105037 000000G     CLR    SLGOLD         ;Clear gold-key flag
20         ;
21         ; Finished
22         ;
23 004750 000207     9$:    RETURN
```

```
1          .SBTTL  FB      -- FB processing
2          ;-----
3          ; Process FB key.
4          ;
5          ; Inputs:
6          ;   R1 = Job index number.
7          ;
8 004752    FB:
9          ;
10         ; See if this is a user-defined key
11         ;
12 004752          KEYCHK  KC$FB      ; See if this is a user-defined key
13 004760 103404    BCS      9$      ; Br if user-defined key
14         ;
15         ; This is not a user-defined key.
16         ; Invalid key.
17         ;
18 004762 004737 011154'    CALL  RRGBEL      ; Ring the bell
19 004766 105037 000000G    CLRBR  SLGOLD     ; Clear gold-key flag
20         ;
21         ; Finished
22         ;
23 004772 000207    9$:      RETURN
```

```
1  
2  
3  
4  
5  
6  
7  
8 004774  
9  
10  
11  
12 004774  
13 005002 103404  
14  
15  
16  
17  
18 005004 004737 011154'  
19 005010 105037 000000G  
20  
21  
22  
23 005014 000207
```

```
      .SBTTL  F9      -- F9 processing  
-----  
; Process F9 key.  
;  
; Inputs:  
; R1 = Job index number.  
;  
F9:  
;  
; See if this is a user-defined key  
;  
      KEYCHK  KC$F9      ;See if this is a user-defined key  
      BCS     9$         ;Br if user-defined key  
;  
; This is not a user-defined key.  
; Invalid key.  
;  
      CALL    RNGBEL      ;Ring the bell  
      CLRB    SLGOLD      ;Clear gold-key flag  
;  
; Finished  
;  
9$:    RETURN
```

```
1  
2  
3  
4  
5  
6  
7  
8 005016  
9  
10  
11  
12 005016  
13 005024 103404  
14  
15  
16  
17  
18 005026 004737 011154'  
19 005032 105037 000000G  
20  
21  
22  
23 005036 000207
```

```
      .SBTTL  F10      -- F10 processing  
      -----  
      ; Process F10 key.  
      ;  
      ; Inputs:  
      ; R1 = Job index number.  
      ;  
      F10:  
      ;  
      ; See if this is a user-defined key  
      ;  
      KEYCHK  KC$F10      ;See if this is a user-defined key  
      BCS     9$          ;Br if user-defined key  
      ;  
      ; This is not a user-defined key.  
      ; Invalid key.  
      ;  
      CALL    RNGBEL      ;Ring the bell  
      CLRB    SLGOLD      ;Clear gold-key flag  
      ;  
      ; Finished  
      ;  
      9$:     RETURN
```

```
1          .SBTTL  F11  -- F11 processing
2          ; -----
3          ; Process F11 key.
4          ;
5          ; Inputs:
6          ;   R1 = Job index number.
7          ;
8 005040    F11:
9          ;
10         ; See if this is a user-defined key
11         ;
12 005040    KEYCHK  KC$F11          ;See if this is a user-defined key
13 005046    BCS     9$             ;Br if user-defined key
14         ;
15         ; This is not a user-defined key.
16         ; Invalid key.
17         ;
18 005050    004737  011154'        CALL  RNGBEL          ;Ring the bell
19 005054    105037  000000G        CLRB   SLGOLD         ;Clear gold-key flag
20         ;
21         ; Finished
22         ;
23 005060    000207          9$:    RETURN
```

```

1
2
3
4
5
6
7
8 005062
9
10
11
12 005062
13 005070 103414
14
15
16
17
18 005072 105737 000000G
19 005076 001005
20
21
22
23 005100 112705 000010
24 005104 004737 006156'
25 005110 000404
26
27
28
29 005112 105037 000000G
30 005116 004737 011154'
31
32
33
34 005122 000207

```

```

.SBTTL F12 -- F12 processing
-----
; Process F12 key -- BS.
;
; Inputs:
; R1 = Job index number.
;
F12:
; See if this is a user-defined key
;
; KEYCHK KC$F12 ;See if this is a user-defined key
; BCS 9$ ;Br if user-defined key
;
; This is not a user-defined key.
; See if gold key was pressed
;
; TSTB SLGOLD ;Was gold key pressed?
; BNE 5$ ;Br if yes
;
; Gold key was not pressed
;
; MOVB #10,R5 ;Get backspace character
; CALL ICPBS ;Process backspace character
; BR 9$
;
; Gold key was pressed
;
5$: CLRB SLGOLD ;Reset gold key
; CALL RNGBEL ;Ring the bell
;
; Finished
;
9$: RETURN

```

```

1          .SBTTL  F13  -- F13 processing
2          ;-----
3          ; Process F13 key -- LF.
4          ;
5          ; Inputs:
6          ; R1 = Job index number.
7          ;
8 005124    F13:
9          ;
10         ; See if this is a user-defined key
11         ;
12         KEYCHK  KC#F13          ; See if this is a user-defined key
13 005132    103414             BCS  9$          ; Br if user-defined key
14         ;
15         ; This is not a user-defined key.
16         ; See if gold key was pressed
17         ;
18 005134    105737    000000G   TSTB  SLGOLD          ; Was gold key pressed?
19 005140    001005             BNE  5$          ; Br if yes
20         ;
21         ; Gold key was not pressed
22         ;
23 005142    012705    000012     MOV  #12,R5          ; Get line-feed character
24 005146    004737    005764'    CALL  ICPLF          ; Process line-feed character
25 005152    000404             BR    9$
26         ;
27         ; Gold key was pressed
28         ;
29 005154    105037    000000G   5$: CLRB  SLGOLD          ; Reset gold key
30 005160    004737    011154'    CALL  RINGBEL        ; Ring the bell
31         ;
32         ; Finished
33         ;
34 005164    000207             9$:  RETURN

```

```
1          .SBTTL  F14      -- F14 processing
2          ;-----
3          ; Process F14 key.
4          ;
5          ; Inputs:
6          ; R1 = Job index number.
7          ;
8 005166    F14:
9          ;
10         ; See if this is a user-defined key
11         ;
12 005166    KEYCHK  KC$F14      ;See if this is a user-defined key
13 005174    BCS     9$          ;Br if user-defined key
14         ;
15         ; This is not a user-defined key.
16         ; Invalid key.
17         ;
18 005176    004737  011154'    CALL  RRGBEL      ;Ring the bell
19 005202    105037  0000000    CLR   SLGOLD     ;Clear gold-key flag
20         ;
21         ; Finished
22         ;
23 005206    000207    9$:      RETURN
```

```
1          .SBTTL  F15      -- F15 processing
2          ;-----
3          ; Process F15 key -- Help.
4          ;
5          ; Inputs:
6          ;   R1 = Job index number.
7          ;
8 005210    F15:
9          ;
10         ; See if this is a user-defined key
11         ;
12 005210    KEYCHK  KC$F15      ;See if this is a user-defined key
13 005216    BCS     9$         ;Br if user-defined key
14         ;
15         ; This is not a user-defined key.
16         ; Invalid key.
17         ;
18 005220    004737  011154'    CALL  RNGBEL      ;Ring the bell
19 005224    105037  000000G    CLRB   SLGOLD     ;Clear gold-key flag
20         ;
21         ; Finished
22         ;
23 005230    000207    9$:     RETURN
```

```
1          .SBTTL  F16      -- F16 processing
2          ;-----
3          ; Process F16 key -- Do.
4          ;
5          ; Inputs:
6          ;   R1 = Job index number.
7          ;
8 005232    F16:
9          ;
10         ; See if this is a user-defined key
11         ;
12 005232    KEYCHK  KC$F16      ;See if this is a user-defined key
13 005240    BCS     9$          ;Br if user-defined key
14         ;
15         ; This is not a user-defined key.
16         ; Invalid key.
17         ;
18 005242    004737  011154'    CALL  RNGBEL      ;Ring the bell
19 005246    105037  000000G    CLRB   SLGOLD     ;Clear gold-key flag
20         ;
21         ; Finished
22         ;
23 005252    000207    9$:      RETURN
```

```
1          .SBTTL  F17      -- F17 processing
2          ;-----
3          ; Process F17 key.
4          ;
5          ; Inputs:
6          ;   R1 = Job index number.
7          ;
8 005254    F17:
9          ;
10         ; See if this is a user-defined key
11         ;
12 005254    KEYCHK  KC$F17      ;See if this is a user-defined key
13 005262    BCS     9$          ;Br if user-defined key
14         ;
15         ; This is not a user-defined key.
16         ; Invalid key.
17         ;
18 005264    004737  011154'    CALL  RNGBEL      ;Ring the bell
19 005270    105037  000000G    CLRB   SLGOLD     ;Clear gold-key flag
20         ;
21         ; Finished
22         ;
23 005274    000207    9$:      RETURN
```

```
1          .SBTTL  F18      -- F18 processing
2          ;-----
3          ; Process F18 key.
4          ;
5          ; Inputs:
6          ;   R1 = Job index number.
7          ;
8 005276    F18:
9          ;
10         ; See if this is a user-defined key
11         ;
12 005276    KEYCHK  KC$F18      ;See if this is a user-defined key
13 005304    BCS     9$          ;Br if user-defined key
14         ;
15         ; This is not a user-defined key.
16         ; Invalid key.
17         ;
18 005306    004737  011154'    CALL  RNGBEL      ;Ring the bell
19 005312    105037  000000G    CLR    SLGOLD     ;Clear gold-key flag
20         ;
21         ; Finished
22         ;
23 005316    000207          9$:  RETURN
```

```
1          .SBTTL  F19      -- F19 processing
2          ;-----
3          ; Process F19 key.
4          ;
5          ; Inputs:
6          ;   R1 = Job index number.
7          ;
8 005320    F19:
9          ;
10         ; See if this is a user-defined key
11         ;
12 005320    KEYCHK  KC$F19      ; See if this is a user-defined key
13 005326    BCS     9$          ; Br if user-defined key
14         ;
15         ; This is not a user-defined key.
16         ; Invalid key.
17         ;
18 005330    004737  011154'    CALL  RNOBEL      ; Ring the bell
19 005334    105037  000000G    CLRB   SLGOLD     ; Clear gold-key flag
20         ;
21         ; Finished
22         ;
23 005340    000207    9$:      RETURN
```

```
1          .SBTTL  F20      -- F20 processing
2          -----
3          ; Process F20 key.
4          ;
5          ; Inputs:
6          ; R1 = Job index number.
7          ;
8 005342    F20:
9          ;
10         ; See if this is a user-defined key
11         ;
12 005342    KEYCHK  KC$F20          ; See if this is a user-defined key
13 005350    BCS     9$              ; Br if user-defined key
14         ;
15         ; This is not a user-defined key.
16         ; Invalid key.
17         ;
18 005352    004737  011154'        CALL  RNGBEL          ; Ring the bell
19 005356    105037  000000G        CLR   SLGOLD         ; Clear gold-key flag
20         ;
21         ; Finished
22         ;
23 005362    000207          9$:    RETURN
```

KEYDOT -- Key "." processing

```
1          .SBTTL  KEYDOT -- Key "." processing
2          ;-----
3          ; Period key -- No defined function
4          ;
5 005364    KEYDOT:
6          ;
7          ; See if this is a user-defined key
8          ;
9 005364    KEYCHK  KC$DOT      ;See if this is a user-defined key
10 005372 103404    BCS      9$      ;Br if user-defined key
11          ;
12          ; This is not a user-defined key.
13          ; Invalid key.
14          ;
15 005374 004737 011154'    CALL  RNGBEL      ;Ring the bell
16 005400 105037 000000G    CLRBR  SLGOLD     ;Clear gold-key flag
17          ;
18          ; Finished
19          ;
20 005404 000207    9$:      RETURN
```

DOCTRL -- Process control characters

```

1          .SBTTL DOCTRL -- Process control characters
2          ;-----
3          ; DOCTRL is called from the input interrupt character processing when
4          ; we determine that the character being processed is a control character.
5          ;
6          ; Inputs:
7          ;   R1 = Virtual line number.
8          ;   R5 = Character to process.
9          ;
10         DOCTRL: MOV     R2, -(SP)
11         ;
12         ; See if this is a request to print the current window
13         ;
14         005406 010246          CMPB     R5, VVPWCH      ; Request to print current window?
15         005414 001013          BNE      5$             ; Br if not
16         005416 032761 000000G 000000G BIT      ##PWKEY, LSW11(R1); Is print window control char enabled?
17         005424 001407          BEQ      5$             ; Br if not
18         005426 016102 000000G          MOV      LWINDO(R1), R2 ; Is windowing enabled for this process?
19         005432 001404          BEQ      5$             ; Br if not
20         ;
21         ; Request to print the current screen window
22         ;
23         005434          OCALL    WINPRT      ; Print the window
24         005442 000425          BR       9$
25         ;
26         ; See if this is a request to switch to a virtual line
27         ;
28         005444 120537 000000G 5$:    CMPB     R5, VVLSCH      ; Is this char a request to switch to vir line?
29         005450 001013          BNE      1$             ; Br if not
30         005452 032761 000000G 000000G BIT      ##CTRLW, LSW3(R1); Was last char also ctrl-W?
31         005460 001004          BNE      2$             ; Br if yes -- Treat two ctrl-W like one ctrl-W
32         005462 052761 000000G 000000G BIS      ##CTRLW, LSW3(R1); Remember last character was ctrl-W
33         005470 000412          BR       9$             ; Finished with this character
34         005472 042761 000000G 000000G 2$:    BIC      ##1ESC, LSW(R1) ; Say last char was not escape
35         005500 042761 000000G 000000G 1$:    BIC      ##CTRLW, LSW3(R1); Say last char not ctrl-W
36         ;
37         ; This is an ordinary control character
38         ;
39         005506 010500          MOV      R5, R0          ; Get the control character
40         005510 006300          ASL      R0              ; Convert to word table index
41         005512 004770 005522'    CALL     @CTLRTN(R0)    ; Call appropriate processing routine
42         ;
43         ; Finished
44         ;
45         005516 012602 9$:    MOV      (SP)+, R2
46         005520 000207          RETURN
47         ;-----
48         ; Branch table for control character processing routines.
49         ;
50         ;
51         005522 007040' CTLRTN: .WORD    ICPNUL          ; 00 - NUL
52         005524 006376'          .WORD    ICPCTA          ; 01 - SOH (control-A)
53         005526 000614'          .WORD    REGCHR          ; 02 - STX
54         005530 006422'          .WORD    ICPCTC          ; 03 - ETX (control-C)
55         005532 000614'          .WORD    REGCHR          ; 04 - EDT
56         005534 000614'          .WORD    REGCHR          ; 05 - ENQ
57         005536 000614'          .WORD    REGCHR          ; 06 - ACK

```

DOCTRL -- Process control characters

58 005540 000614'	. WORD REGCHR	; 07 - BEL
59 005542 006156'	. WORD ICPBS	; 10 - Backspace
60 005544 000614'	. WORD REGCHR	; 11 - TAB
61 005546 005764'	. WORD ICPLF	; 12 - Line feed
62 005550 000614'	. WORD REGCHR	; 13 - VT
63 005552 000614'	. WORD REGCHR	; 14 - FF
64 005554 005622'	. WORD ICPCR	; 15 - Carriage return
65 005556 000614'	. WORD REGCHR	; 16 - SO
66 005560 000614'	. WORD REGCHR	; 17 - SI (control-O)
67 005562 000614'	. WORD REGCHR	; 20 - DLE
68 005564 000614'	. WORD REGCHR	; 21 - DC1 (control-Q)
69 005566 006524'	. WORD ICPCTR	; 22 - DC2 (control-R)
70 005570 000614'	. WORD REGCHR	; 23 - DC3 (control-S)
71 005572 000614'	. WORD REGCHR	; 24 - DC4
72 005574 006560'	. WORD ICPCTU	; 25 - NAK (control-U)
73 005576 000614'	. WORD REGCHR	; 26 - SYN
74 005600 006524'	. WORD ICPCTR	; 27 - ETB (control-W)
75 005602 000614'	. WORD REGCHR	; 30 - CAN (control-X)
76 005604 000614'	. WORD REGCHR	; 31 - EM
77 005606 006700'	. WORD ICPCTZ	; 32 - SUB (control-Z)
78 005610 006274'	. WORD ICPESC	; 33 - ESC
79 005612 000614'	. WORD REGCHR	; 34 - FS
80 005614 000614'	. WORD REGCHR	; 35 - GS
81 005616 000614'	. WORD REGCHR	; 36 - RS
82 005620 000614'	. WORD REGCHR	; 37 - US

ICPCR -- Carriage-return processing

```

1          .SBTTL  ICPCR  -- Carriage-return processing
2          -----
3          ; Process Carriage-return character.
4          ;
5          ; Inputs:
6          ;   R1 = Virtual line index number.
7          ;   R5 = Current input character.
8          ;
9 005622    010446  ICPCR:  MOV      R4, -(SP)
10         ;
11         ; If we have not yet gotten line-feed character, set flag and wait
12         ; for line-feed to arrive.
13         ;
14 005624    105737  000000G      TSTB     SLCR          ; Have we seen line-feed yet?
15 005630    001003          BNE      5$          ; Br if yes
16         ;
17         ; We have not seen line-feed yet. Set flag and wait for line-feed.
18         ;
19 005632    105237  000000G      INCB     SLCR          ; Set flag saying we are waiting for line feed
20 005636    000450          BR       9$          ; Defer processing until after line feed
21         ;
22         ; We have received carriage return and line feed.
23         ; See if Gold key has been pressed.
24         ;
25 005640    105037  000000G  5$:      CLRB     SLCR          ; Clear carriage return flag
26 005644    105737  000000G          TSTB     SLGOLD        ; Has gold key been pressed?
27 005650    001407          BEQ      1$          ; Br if not
28         ;
29         ; Gold key was pressed -- Truncate line before terminating it
30         ;
31 005652    105037  000000G          CLRB     SLGOLD        ; Clear gold-key flag
32 005656    013704  000000G          MOV      SLCX, R4      ; Get current cursor pointer
33 005662    105014          CLRB     (R4)          ; Truncate edit line
34 005664    004737  010336'          CALL     PAINT        ; Redisplay line
35         ;
36         ; Terminate input line.
37         ;
38 005670    004737  007172'  1$:      CALL     SAVLIN        ; Save the completed input line
39 005674    013704  000000G          MOV      SLCX, R4      ; Point to cursor position
40 005700    105724  2$:      TSTB     (R4)+          ; Search for end of line
41 005702    001376          BNE      2$          ; Loop till null hit
42 005704    005304          DEC       R4            ; Point to null
43 005706    112724  000000G          MOVB     #CR, (R4)+    ; Store carriage-return
44 005712    112724  000000G          MOVB     #LF, (R4)+    ; And line-feed at end of line
45 005716    105014          CLRB     (R4)          ; Put null at end
46         ;
47         ; Echo carriage-return & line-feed to terminal
48         ;
49 005720    112700  000000G          MOVB     #CR, R0      ; Get carriage return
50 005724    004737  011210'          CALL     ECHO        ; Echo carriage return
51 005730    032761  000000G 000000G  BIT      #$NOLF, LSW6(R1) ; Are we suppressing LF echoing after CR?
52 005736    001004          BNE      3$          ; Br if yes
53 005740    112700  000000G          MOVB     #LF, R0      ; Get line-feed character
54 005744    004737  011210'          CALL     ECHO        ; Echo line feed
55         ;
56         ; Signal line completed
57         ;

```

```
58 005750 004737 007346'      3#:      CALL      LINFIN      ;Input line is finished
59                               ;
60                               ; Say that cursor is at left margin
61                               ;
62 005754 105061 0000000      CLR      CLR      LCOL(R1)      ;Say cursor is at left margin
63                               ;
64                               ; Finished
65                               ;
66 005760 012604              9#:      MOV      (SP)+,R4
67 005762 000207              RETURN
```

```

1          .SBTTL  ICPLF  -- Line-feed processing
2          -----
3          ; Process Line-feed input character.
4          ;
5          ; Inputs:
6          ;   R1 = Virtual line index number.
7          ;   R5 = Current input character.
8          ;
9 005764   ICPLF:
10         ;
11         ; If this is a line feed that is part of a carriage-return/line-feed pair
12         ; then go to do carriage return processing.
13         ;
14 005764   105737   000000G      TSTB    SLCR          ;Is this line feed after a carriage return?
15 005770   001314          BNE     ICPCR          ;Br if yes -- Go do carriage return processing
16         ;
17         ; This is a line feed by itself
18         ;
19 005772   010246          MOV     R2,-(SP)
20 005774   010346          MOV     R3,-(SP)
21 005776   010446          MOV     R4,-(SP)
22 006000   010546          MOV     R5,-(SP)
23         ;
24         ; See if Gold key (PF1) was pressed before line-feed
25         ;
26 006002   105737   000000G      TSTB    SLGOLD         ;Was gold key pressed?
27 006006   001407          BEQ     1$              ;Br if not
28         ;
29         ; Gold key was pressed -- Replace previously deleted word.
30         ;
31 006010   012702   000000G      MOV     #SLDBUF,R2      ;Point to delete buffer
32 006014   004737   010140'      CALL    INSERT        ;Replace the deleted word
33 006020   105037   000000G      CLRB    SLGOLD         ;Clear the gold key
34 006024   000447          BR      9$
35         ;
36         ; Gold key was not pressed -- Delete the word to the left of the cursor.
37         ; Begin by deleting any delimiters immediately to left of cursor.
38         ;
39 006026   013704   000000G      1$:   MOV     SLCX,R4      ;Get pointer to character under cursor
40 006032   010405          MOV     R4,R5
41 006034   020527   000000G      CMP     R5,#SLEBUF      ;Are we at left end now?
42 006040   101441          BLOS    9$                  ;Br if yes
43 006042   114500          11$:   MOVB    -(R5),R0        ;Get next character to be deleted
44 006044   004737   011010'      CALL    CHKDLM        ;Is this character a delimiter?
45 006050   103006          BCC     4$                  ;Br if not
46         ;
47         ; Found delimiter to left of cursor.
48         ; Delete all consecutive occurrences of the delimiter.
49         ;
50 006052   020527   000000G      3$:   CMP     R5,#SLEBUF      ;Are we at left end of line now?
51 006056   101413          BLOS    6$                  ;Br if yes
52 006060   120045          CMPB    R0,-(R5)            ;See if there are more occurrences of delim
53 006062   001773          BEQ     3$                  ;Br if yes -- keep deleting
54 006064   005205          INC     R5
55         ;
56         ; We have now deleted any delimiters to the left of the cursor.
57         ; Begin to delete the word to the left of the cursor.

```

ICPLF -- Line-feed processing

```

58
59 006066 020527 000000G      4$:    CMP      R5,#SLEBUF      ;Have we hit left end of line?
60 006072 101405                BLOS      6$              ;Br if yes
61 006074 114500                MOV      -(R5),R0          ;Get next char to delete
62 006076 004737 011010'      CALL      CHKDLM          ;Is this character a delimiter?
63 006102 103371                BCC      4$              ;Br if not
64 006104 005205                INC      R5              ;Don't delete the delimiter
65
66      ; We have now identified the characters to delete.
67      ; Save characters being deleted in deleted-string buffer.
68
69 006106 010503      6$:    MOV      R5,R3              ;Point to left-most character to delete
70 006110 012702 000000G      MOV      #SLDBUF,R2          ;Point to deleted-string buffer
71 006114 112322      10$:   MOV      (R3)+,(R2)+          ;Move chars to holding buffer
72 006116 020304                CMP      R3,R4              ;Moved all that need to be deleted?
73 006120 103775                BLO      10$              ;Loop if not
74 006122 105012                CLRB      (R2)              ;Put null at end of deleted string
75
76      ; Delete the characters from the buffer
77
78 006124 010503      MOV      R5,R3
79 006126 112425      7$:    MOV      (R4)+,(R5)+          ;Move over characters to right of deleted ones
80 006130 001376                BNE      7$
81 006132 010304                MOV      R3,R4              ;Get new cursor character index
82
83      ; Redisplay line
84
85 006134 004737 010336'      CALL      PAINT              ;Redisplay from delete point to right end
86
87      ; Reposition cursor
88
89 006140 004737 010642'      CALL      CHRPOS            ;Reposition cursor
90
91      ; Finished
92
93 006144 012605      9$:    MOV      (SP)+,R5
94 006146 012604                MOV      (SP)+,R4
95 006150 012603                MOV      (SP)+,R3
96 006152 012602                MOV      (SP)+,R2
97 006154 000207                RETURN

```

ICPBS -- Backspace processing

```

1          .SBTTL  ICPBS  -- Backspace processing
2          ;-----
3          ; Process a Backspace character.
4          ;
5          ; Inputs:
6          ;   R1 = Job index number.
7          ;
8 006156 010446 ICPBS:  MOV      R4,-(SP)
9          ;
10         ; See if Gold key was pressed
11         ;
12 006160 013704 0000000 MOV      SLCX,R4      ;Get cursor pointer
13 006164 105737 0000000 TSTB     SLGOLD      ;Was gold key pressed?
14 006170 001423        BEQ      1$      ;Br if not
15         ;
16         ; Gold key was pressed.
17         ; Exchange character under cursor with character to left of cursor.
18         ;
19 006172 105037 0000000 CLR      SLGOLD      ;Clear gold-key flag
20 006176 020427 0000000 CMP      R4,#SLEBUF    ;Is cursor at left end of line?
21 006202 001402        BEQ      3$      ;Br if yes
22 006204 105714        TSTB     (R4)      ;Is cursor at right end of line?
23 006206 001003        BNE      4$      ;Br if not
24 006210 004737 011154' 3$:  CALL     RINGBEL    ;Ring bell
25 006214 000425        BR       9$      ;And ignore command
26 006216 111400        4$:  MOVB     (R4),R0    ;Get character under cursor
27 006220 114464 0000001 MOVB     -(R4),1(R4)  ;Move char to left of cursor to cursor pos
28 006224 110014        MOVB     R0,(R4)      ;Store old char to left of old cursor pos
29 006226 004737 010336' CALL     PAINT      ;Redisplay the line
30 006232 004737 010642' CALL     CHRPOS     ;Position cursor
31 006236 000414        BR       9$
32         ;
33         ; Gold key was not pressed.
34         ; Exchange character under cursor with character to right of cursor.
35         ;
36 006240 112400        1$:  MOVB     (R4)+,R0    ;Get current char under the cursor
37 006242 001762        BEQ      3$      ;Br if we were at right end of line
38 006244 105714        TSTB     (R4)      ;Is next character end of line?
39 006246 001760        BEQ      3$      ;Br if yes
40 006250 111444        MOVB     (R4),-(R4)    ;Move char to right of cursor under cursor
41 006252 110064 0000001 MOVB     R0,1(R4)
42 006256 004737 010336' CALL     PAINT      ;Redisplay the line
43 006262 005204        INC      R4          ;Cursor moves right 1 character
44 006264 004737 010642' CALL     CHRPOS     ;Position cursor
45         ;
46         ; Finished
47         ;
48 006270 012604        9$:  MOV      (SP)+,R4
49 006272 000207        RETURN

```

ICPESC -- Escape processing

```

1          .SBTTL  ICPESC -- Escape processing
2          ;-----
3          ; Process an Escape character.
4          ;
5          ; Inputs:
6          ;   R1 = Job index number.
7          ;   R5 = Escape character.
8          ;
9 006274    ICPESC:
10         ;
11         ; Set address of routine to be called to process 1st character after
12         ; escape.
13         ;
14 006274    012700    001216'      MOV     #EPCH1,R0      ;Routine for 1st char after escape
15 006300    004737    011602'      CALL    CHK52          ;Is this a VT52 terminal?
16 006304    103002                    BCC     1$           ;Br if not
17 006306    012700    001240'      MOV     #EP52,R0      ;Set routine for VT52
18 006312    010037    000000G      1$:  MOV     R0,SLCSR    ;Address of routine to process next char
19 006316    012737    000000G 000000G  MOV     #SLCSBF,SLCSPT ;Set pointer to start of buffer
20         ;
21         ; Finished
22         ;
23 006324    000207                    RETURN
24         ;
25         ;-----
26         ; Received a CSI control character from a VT200 terminal.
27         ;
28         ; Inputs:
29         ;   R1 = Job index number.
30         ;   R5 = character.
31         ;
32 006326    012737    001304' 000000G ICPCSI: MOV     #EPCH2,SLCSR ;Set address of routine to process next char
33 006334    012700    000000G          MOV     #SLCSBF,R0    ;Get pointer to start of buffer
34 006340    112720    000133          MOVB    #'[, (R0)+    ;Store "[" as 1st char of sequence
35 006344    010037    000000G          MOV     R0,SLCSPT    ;Set pointer into control sequence buffer
36 006350    000207                    RETURN
37         ;
38         ;-----
39         ; Received a SS3 control character from a VT200 terminal.
40         ;
41         ; Inputs:
42         ;   R1 = Job index number.
43         ;   R5 = Character.
44         ;
45 006352    012737    001304' 000000G ICPSS3: MOV     #EPCH2,SLCSR ;Set address of routine to process next char
46 006360    012700    000000G          MOV     #SLCSBF,R0    ;Get pointer to start of buffer
47 006364    112720    000117          MOVB    #'O, (R0)+    ;Store "O" as 1st char of sequence
48 006370    010037    000000G          MOV     R0,SLCSPT    ;Set pointer into control sequence buffer
49 006374    000207                    RETURN

```

ICPCTA -- Control-A processing

```
1          .SBTTL  ICPCTA -- Control-A processing
2          ;-----
3          ; Control-A -- toggle overstrike/insert mode
4          ;
5 006376 105737 000000G ICPCTA: TSTB  SLOVER      ;In overstrike mode now?
6 006402 001004          BNE    1$          ;Br if yes
7 006404 112737 000001 000000G      MOVB  #1,SLOVER ;Enter overstrike mode
8 006412 000402          BR     9$
9 006414 105037 000000G      1$:  CLRB  SLOVER      ;Exit overstrike mode
10 006420 000207          9$:  RETURN
```

ICPCTC -- Control-C processing

```

1          .SBTTL  ICPCTC -- Control-C processing
2          ;-----
3          ; Process a control-C input character.
4          ;
5          ; Inputs:
6          ;   R1 = Virtual line index number.
7          ;   R5 = Current input character.
8          ;
9 006422    ICPCTC:
10         ;
11         ; If line is a .SCCA, treat control-C like control-Z
12         ;
13 006422    005761 000000G          TST      LSCCA(R1)          ; Did program do a .SCCA?
14 006426    001402          BEQ      3$              ; Br if not
15 006430    000137 006700'          JMP      ICPCTZ          ; Treat control-C like control-Z
16 006434    032761 000000G 000000G 3$: bit    $$scca,lsw5(r1) ; Suppressing control-c abort (RUN/SCCA)
17 006442    001401          beq      2$              ; BR if no
18 006444    000207          return
19
20         ;
21         ; Clear edit buffer
22         ;
23 006446    012704 000000G 2$:   MOV      #SLEBUF,R4          ; Point to front of edit buffer
24 006452    105014          CLRB     (R4)              ; Say buffer is empty
25 006454    004737 010336'          CALL    PAINT          ; Clear line on screen
26
27         ; Echo "^C" to terminal
28         ;
29 006460    004737 011130'          CALL    ECOCTL          ; Echo "^C"
30
31         ; Send "^C" to log file if we are doing logging
32         ;
33 006464    032737 000000G 000000G BIT      #LF$IN,LOGFLG    ; Are we logging input characters?
34 006472    001407          BEQ      1$              ; Br if not
35 006474    110500          MOVB     R5,R0            ; Get control-C character
36 006476          DCALL    LOGCHR          ; Log Control-C
37 006504          DCALL    LOGCR          ; Log CR-LF
38
39         ; Stop program execution and enter Kmon
40         ;
41 006512    042761 000000G 000000G 1$: BIC      $$SLINI,LSW7(R1); Say SL must reinitialize for next line
42 006520    004737 000000G          CALL    STOP

```

ICPCTR -- Control-R processing

```
1          .SBTTL  ICPCTR -- Control-R processing
2          ;-----
3          ; Process control-R input character.
4          ;
5          ICPCTR: MOV      R2,-(SP)
6          MOV      R4,-(SP)
7          ;
8          ; Save current cursor position and redisplay the line
9          ;
10         MOV      SLCX,R2          ; Save current cursor character pointer
11         MOV      #SLEBUF,R4      ; Redisplay entire line
12         CALL     PAINT           ; Do the display
13         ;
14         ; Reposition cursor
15         ;
16         MOV      R2,R4           ; Get original cursor pointer
17         CALL     CHRPOS          ; Reposition cursor
18         ;
19         ; Finished
20         ;
21         MOV      (SP)+,R4
22         MOV      (SP)+,R2
23         RETURN
```

```

1          .SBTTL  ICPCTU -- Control-U processing
2          ;-----
3          ; Process a Control-U character.
4          ;
5          ; Inputs:
6          ;   R1 = Virtual line index number.
7          ;   R5 = Current input character.
8          ;
9 006560    010246    ICPCTU: MOV      R2,-(SP)
10 006562    010346    MOV      R3,-(SP)
11 006564    010446    MOV      R4,-(SP)
12          ;
13          ; See if Gold key (PF1) was pressed before control-U
14          ;
15 006566    105737    000000G    TSTB     SLGOLD      ;Was Gold key pressed?
16 006572    001407    BEQ      3$      ;Br if not
17          ;
18          ; Gold key was pressed -- Put back previously deleted text
19          ;
20 006574    012702    000000G    MOV      #SLDBUF,R2      ;Point to buffer with deleted text
21 006600    004737    010140'    CALL     INSERT      ;Insert in front of cursor
22 006604    105037    000000G    CLRB     SLGOLD      ;Clear gold-key flag
23 006610    000427    BR       9$      ;Finished
24          ;
25          ; Gold key was not pressed -- Delete all characters to left of cursor
26          ; Check to see if there is anything to delete.
27          ;
28 006612    013704    000000G    3$:     MOV      SLCX,R4      ;Get pointer to char under cursor
29 006616    020427    000000G    CMP      R4,#SLEBUF      ;Is cursor at left end now?
30 006622    001422    BEQ      9$      ;Br if yes -- nothing to delete
31          ;
32          ; Move deleted characters to holding buffer
33          ;
34 006624    012702    000000G    MOV      #SLEBUF,R2      ;Point to regular edit buffer
35 006630    012703    000000G    MOV      #SLDBUF,R3      ;Point to deleted-characters buffer
36 006634    112223    1$:     MOVB     (R2)+,(R3)+      ;Save characters being deleted
37 006636    020204    CMP      R2,R4      ;Have we saved all?
38 006640    103775    BLO      1$      ;Br if not
39 006642    105013    CLRB     (R3)      ;Put null at end of delete string
40          ;
41          ; Delete characters from the buffer
42          ;
43 006644    012702    000000G    MOV      #SLEBUF,R2      ;Point to start of string to delete
44 006650    112422    2$:     MOVB     (R4)+,(R2)+      ;Move characters left over deleted ones
45 006652    001376    BNE      2$      ;Loop till all moved
46          ;
47          ; Redisplay the line
48          ;
49 006654    012704    000000G    MOV      #SLEBUF,R4      ;Redisplay entire line
50 006660    004737    010336'    CALL     PAINT
51          ;
52          ; Reposition cursor
53          ;
54 006664    004737    010642'    CALL     CHRPOS      ;Position cursor
55          ;
56          ; Finished
57          ;

```

58	006670	012604	9\$:	MOV	(SP)+, R4
59	006672	012603		MOV	(SP)+, R3
60	006674	012602		MOV	(SP)+, R2
61	006676	000207		RETURN	

```

1
2
3
4
5
6
7
8
9 006700 010446
10
11
12
13 006702 013704 000000G
14 006706 105724
15 006710 001376
16 006712 005304
17 006714 004737 010642'
18
19
20
21 006720 004737 011130'
22
23
24
25 006724 112700 000000G
26 006730 004737 011226'
27 006734 112700 000000G
28 006740 004737 011226'
29
30
31
32 006744 004737 007172'
33
34
35
36 006750 110524
37 006752 105014
38
39
40
41 006754 004737 007346'
42
43
44
45 006760 012604
46 006762 000207

      .SBTTL  ICPCTZ -- Control-Z processing
      -----
      ; Process Control-Z character.
      ;
      ; Inputs:
      ;   R1 = Virtual line index number.
      ;   R5 = Current input character.
      ;
      ICPCTZ: MOV      R4, -(SP)
      ;
      ; Move cursor to end of line
      ;
      1$:      MOV      SLCX, R4      ; Get current cursor pointer
      2$:      TSTB     (R4)+        ; Search for null at end of line
              BNE      2$           ; Loop till null hit
              DEC      R4           ; Point to null
              CALL     CHRPOS       ; Position cursor to end of line
      ;
      ; Echo "^Z" to terminal
      ;
              CALL     ECOCTL       ; Echo "^Z"
      ;
      ; Echo carriage return / line feed
      ;
              MOVB     #CR, R0      ; Get carriage return
              CALL     ECHO2        ; Echo it
              MOVB     #LF, R0      ; Get line feed
              CALL     ECHO2        ; Echo it
      ;
      ; Save input line up to (but not including) control-Z
      ;
              CALL     SAVLIN       ; Save input line
      ;
      ; Insert control-Z into buffer
      ;
              MOVB     R5, (R4)+    ; Insert control-Z into buffer
              CLRB     (R4)         ; Put null at end of buffer
      ;
      ; We have finished input line
      ;
              CALL     LINFIN       ; Say we have finished input line
      ;
      ; Finished
      ;
              MOV      (SP)+, R4
              RETURN
  
```

ICPRUB -- Rubout processing

```

1          .SBTTL  ICPRUB -- Rubout processing
2          ;-----
3          ; Process a rubout character.
4          ;
5 006764    010246  ICPRUB: MOV      R2,-(SP)
6          ;
7          ; See if Gold key (PF1) was pressed before rubout.
8          ;
9 006766    105737    000000G      TSTB      SLGOLD      ;Was Gold key pressed?
10 006772    001407      BEQ      1$      ;Br if not
11          ;
12          ; Gold key was pressed -- Put back previously deleted character
13          ;
14 006774    012702    000000G      MOV      #SLCBUF,R2      ;Point to deleted-character buffer
15 007000    004737    010140'      CALL      INSERT      ;Insert in line
16 007004    105037    000000G      CLRB      SLGOLD      ;Clear the Gold-key flag
17 007010    000411      BR       9$      ;Finished
18          ;
19          ; Gold key was not pressed -- Delete character to left of cursor
20          ;
21 007012    013702    000000G  1$:  MOV      SLCX,R2      ;Get pointer to character under cursor
22 007016    020227    000000G      CMP      R2,#SLEBUF      ;Is cursor at left end of line?
23 007022    101404      BLOS     9$      ;Br if yes
24          ;
25          ; Save character being deleted and then delete it
26          ;
27 007024    114237    000000G  2$:  MOVB     -(R2),SLCBUF      ;Save character being deleted
28 007030    004737    007454'      CALL      DELLFT      ;Delete character to left of cursor
29          ;
30          ; Finished
31          ;
32 007034    012602      9$:  MOV      (SP)+,R2
33 007036    000207      RETURN

```

ICPNUL -- Null processing

1 .SBTTL ICPNUL -- Null processing

2

3

4

5 007040 000207

;-----
; Process a null character

; ICPNUL: RETURN

INWAIT -- Wait for input characters

```

1          .SBTTL  INWAIT -- Wait for input characters
2          ;-----
3          ; INWAIT is called to wait for input characters.
4          ;
5          ; Inputs:
6          ; R1 = Job index number
7          ;
8 007042    INWAIT:
9          ;
10         ; If we previously suspended input from silo buffer to prevent
11         ; input buffer overrun, reenale input now.
12         ;
13 007042    042761  000000G 000000G          BIC    ##XSTOP,LSW6(R1);Reenable input from silo buffer
14 007050    052761  000000G 000000G          BIS    ##NDICP,LSW10(R1);Say line needs input character servicing
15 007056    005237  000000G          INC    NEDCDI          ;Say input character processing needed
16         ;
17         ; Suspend job until activation character is received.
18         ;
19 007062    1$:    OCALL  SIGWAT          ;Signal virtual line wait condition
20 007070    042761  000000G 000000G          BIC    ##NDIN,LSW3(R1) ;Allow input to be accepted for line
21 007076    042761  000000G 000000G          BIC    ##SUCF,LSW9(R1) ;No longer in startup command file
22 007104    004737  000000G          2$:    CALL  CHKABT          ;See if job has been aborted
23 007110          DISABL          ;;;** Disable **
24 007116    005761  000000G          TST    LACTIV(R1)          ;;;Got any activation chars yet?
25 007122    001017          BNE    3$          ;;;Br if yes
26 007124    032761  000000G 000000G          BIT    ##NOINT,LSW7(R1);;;Is program being run non-interactively?
27 007132    001006          BNE    4$          ;;;Br if yes
28 007134    013761  000000G 000000G          MOV    VINTIO,LHIPCT(R1);;;Reset high-priority hit limit for job
29 007142    013761  000000G 000000G          MOV    VQUAN1,LITIME(R1);;;Reset interactive CPU time limit
30 007150    012700  000000G          4$:    MOV    #S$INWT,RO          ;;;Get waiting-for-input state
31 007154    004737  000000G          CALL  QHDSPN          ;;;Suspend job and wait for activation char
32 007160    000751          BR     2$          ;Go check again
33         ;
34         ; There are input characters available
35         ;
36 007162    3$:    ENABL          ;;;** Enable **
37         ;
38         ; Finished
39         ;
40 007170    000207          RETURN

```

SAVLIN -- Save current input line

```

1          .SBTTL  SAVLIN -- Save current input line
2          -----
3          ; SAVLIN is called to save the current input line as the "last line"
4          ; which can be recalled later by use of the up-arrow key.
5          ;
6 007172    010246 SAVLIN: MOV      R2,-(SP)
7 007174    010346      MOV      R3,-(SP)
8          ;
9          ; If line is null, there is nothing to save
10         ;
11 007176    012702 000000G      MOV      #SLEBUF,R2      ;Point to current line buffer
12 007202    105712      TSTB      (R2)      ;Is current line null?
13 007204    001450      BEQ       9$          ;Br if yes -- Don't save a null line
14         ;
15         ; If current line matches last line, don't push saved lines
16         ;
17 007206    013703 000000G      MOV      SLSPTR,R3      ;Point to most recently saved line
18 007212    020327 000000G 4$:    CMP      R3,#SLEND      ;Gone past end of save buffer?
19 007216    103402      BLO       6$          ;Br if not
20 007220    012703 000000G      MOV      #SLLBUF,R3      ;Wrap to top of buffer
21 007224    121223 6$:    CMPB     (R2),(R3)+      ;Does next char in cur line match saved line?
22 007226    001003      BNE       5$          ;Br if not
23 007230    105722      TSTB     (R2)+      ;Have we reached end of current line?
24 007232    001367      BNE       4$          ;Loop if not
25 007234    000434      BR        9$          ;Line is same as last saved, don't save again
26         ;
27         ; We want to save the current line in the save buffer.
28         ; Copy to save buffer above previous saved line.
29         ;
30 007236    105722 5$:    TSTB     (R2)+      ;Find null at end of current command
31 007240    001376      BNE       5$
32 007242    013703 000000G      MOV      SLSPTR,R3      ;Point to start of last saved line
33 007246    020327 000000G 8$:    CMP      R3,#SLLBUF      ;Gone past top of save buffer?
34 007252    101002      BHI       10$         ;Br if not
35 007254    012703 000000G      MOV      #SLEND,R3      ;Wrap around to end of save buffer
36 007260    114243 10$:   MOVB     -(R2),-(R3)      ;Copy new line into save buffer
37 007262    020227 000000G      CMP      R2,#SLEBUF      ;Copied all of new line?
38 007266    101367      BHI       8$          ;Br if not
39 007270    010337 000000G      MOV      R3,SLSPTR      ;Save pointer to start of most recent line
40         ;
41         ; Now null out the remainder of any command we may have partially covered
42         ;
43 007274    020327 000000G 11$:   CMP      R3,#SLLBUF      ;Hit top of save buffer?
44 007300    101002      BHI       12$         ;Br if not
45 007302    012703 000000G      MOV      #SLEND,R3      ;Wrap around to end of save buffer
46 007306    105743 12$:   TSTB     -(R3)          ;Reached null that terminates prev line?
47 007310    001402      BEQ       13$         ;Br if yes
48 007312    105013      CLRB     (R3)          ;Null out remainder of saved line
49 007314    000767      BR        11$
50         ;
51         ; Reset recall cycle
52         ;
53 007316    005037 000000G 13$:   CLR      SLCYC1      ;Say no recall cycle active
54 007322    005037 000000G      CLR      SLCYC2
55         ;
56         ; Set recall pointer to recall last saved line
57         ;

```

SAVLIN -- Save current input line

```
58 007326 013737 0000000 0000000 9#:      MOV      SLSPTR, SLLPTR      ;Set ptr to next command to recall
59 007334 105037 0000000                   CLRB      SLDOWN          ;Say down-arrow was not last command
60                                           ;
61                                           ; Finished
62                                           ;
63 007340 012603                   MOV      (SP)+, R3
64 007342 012602                   MOV      (SP)+, R2
65 007344 000207                   RETURN
```

LINFIN -- Terminate input line

```

1          .SBTTL  LINFIN -- Terminate input line
2          ;-----
3          ; LINFIN is called when an input line is completely acquired.
4          ;
5          ; Inputs:
6          ;   R1 = Job index number.
7          ;
8 007346    010246    LINFIN: MOV      R2,-(SP)
9 007350    010346          MOV      R3,-(SP)
10         ;
11         ; Make sure job priority is not higher than normal (non-single character
12         ; activation) input complete.
13         ;
14 007352    026127    000000G 000000G    CMP      LSTATE(R1),#S$TTFN; Is prio higher than normal TT input?
15 007360    103004          BHIS     3$          ;Br if not
16 007362    012700    000000G          MOV      #S$TTFN,R0          ;Lower priority of this job
17 007366    004737    000000G          CALL     ENQTL          ;Requeue at tail of S$TTFN state queue
18         ;
19         ; If terminal logging is wanted, log the line now
20         ;
21 007372    032737    000000G 000000G 3$:    BIT      #LF$IN,LOGFLG    ;Are we logging input characters?
22 007400    001410          BEQ       5$          ;Br if not
23 007402    012702    000000G          MOV      #SLEBUF,R2          ;Point to line buffer
24 007406    112200          4$:    MOVB     (R2)+,R0          ;Get next character from the line
25 007410    001404          BEQ       5$          ;Br when null hit
26 007412          OCALL     LOGCHR          ;Log the character
27 007420    000772          BR        4$
28         ;
29         ; Set up pointer to character string to be passed to program
30         ;
31 007422    012737    000000G 000000G 5$:    MOV      #SLEBUF,SLOPTR    ;Set pointer for GTSLCH routine
32         ;
33         ; Set cursor position at end of line
34         ;
35 007430    113761    000000G 000000G          MOVB     SLECOL,LCOL(R1) ;Remember cursor position at end of line
36         ;
37         ; Clear field width activation and field width limit values
38         ;
39 007436    005061    000000G          CLR      LFWLIM(R1)          ;No field width limit now
40 007442    005061    000000G          CLR      LAFSIZ(R1)          ;No field width activation now
41         ;
42         ; Finished
43         ;
44 007446    012603          MOV      (SP)+,R3
45 007450    012602          MOV      (SP)+,R2
46 007452    000207          RETURN

```

1			.SBTTL DELLFT -- Delete character to left of cursor
2			-----
3			; DELLFT is called to delete the character to the left of the current
4			; cursor position and redisplay any characters to the right of the
5			; deleted character.
6			; The cursor is left positioned on the character that was to the right
7			; of the deleted character.
8			;
9			; Inputs:
10			; SLCX = Pointer to character under the cursor.
11			; SLCCOL = Column number where cursor is positioned.
12			;
13	007454	010346	DELLFT: MOV R3, -(SP)
14	007456	010446	MOV R4, -(SP)
15			;
16			; If cursor is already at left-most character then there is nothing
17			; to do.
18			;
19	007460	013704 000000G	MOV SLCX, R4 ;Get pointer to character under cursor
20	007464	020427 000000G	CMP R4, #SLEBUF ;Is cursor at left end?
21	007470	001413	BEG 9\$;Br if at left-mode char -- Nothing to delete
22			;
23			; Remove deleted character from buffer
24			;
25	007472	005304	DEC R4 ;Point to the deleted character
26	007474	111437 000000G	MOVB (R4), SLCBUF ;Save deleted character
27	007500	010403	MOV R4, R3
28	007502	116323 0000001	1\$: MOVB 1(R3), (R3)+ ;Move chars left over deleted char
29	007506	001375	BNE 1\$
30			;
31			; Redisplay any characters to the right of the deleted character
32			;
33	007510	004737 010336'	CALL PAINT ;Redisplay chars to left of deleted char
34			;
35			; Reposition cursor to character that was to right of one deleted.
36			;
37	007514	004737 010642'	CALL CHRPOS ;Reposition cursor
38			;
39			; Finished
40			;
41	007520	012604	9\$: MOV (SP)+, R4
42	007522	012603	MOV (SP)+, R3
43	007524	000207	RETURN

```

1          .SBTTL  SLMVUP -- Move up to previous stored command
2          -----
3          ; This routine returns a pointer to the previous saved command moving
4          ; in an up-arrow direction.
5          ;
6          ; Inputs:
7          ;   R5 = Pointer to current command.
8          ;
9          ; Outputs:
10         ;   R5 = Pointer to previous command.
11         ;
12 007526 010446 SLMVUP: MOV      R4, -(SP)
13         ;
14         ; Skip up to the null at the end of the current command
15         ;
16 007530 010504         MOV      R5, R4
17 007532 020427 0000000 1$:      CMP      R4, #SLEND      ; Hit end of buffer?
18 007536 103402         BLO      2$                      ; Br if not
19 007540 012704 0000000         MOV      #SLLBUF, R4      ; Wrap around to the top
20 007544 105724 2$:      TSTB     (R4)+                  ; Reached null at end of saved command?
21 007546 001371         BNE      1$                      ; Br if not
22         ;
23         ; Now skip over nulls at the end of the command
24         ;
25 007550 020427 0000000 3$:      CMP      R4, #SLEND      ; Hit end of buffer?
26 007554 103402         BLO      4$                      ; Br if not
27 007556 012704 0000000         MOV      #SLLBUF, R4      ; Wrap around to the top
28 007562 020405 4$:      CMP      R4, R5                  ; Wrapped around to current cmd?
29 007564 001411         BEQ      9$                      ; Br if nothing to recall
30 007566 105724         TSTB     (R4)+                  ; More nulls to skip?
31 007570 001767         BEQ      3$                      ; Br if so
32         ;
33         ; Point to 1st character of command
34         ;
35 007572 020427 0000000 5$:      CMP      R4, #SLLBUF      ; At top of buffer now?
36 007576 101002         BHI      6$                      ; Br if not
37 007600 012704 0000000         MOV      #SLEND, R4      ; Wrap around to bottom
38 007604 005304 6$:      DEC      R4                      ; Point to 1st char of next command
39         ;
40         ; Finished
41         ;
42 007606 010405         MOV      R4, R5                  ; Return pointer in R5
43 007610 012604 7$:      MOV      (SP)+, R4
44 007612 000207         RETURN

```

SLMVDN -- Move down to previous stored command

```

1          .SBTTL  SLMVDN -- Move down to previous stored command
2          ;-----
3          ; This routine returns a pointer to the previous saved command moving
4          ; in a down-arrow direction.
5          ;
6          ; Inputs:
7          ;   R5 = Pointer to current command.
8          ;
9          ; Outputs:
10         ;   R5 = Pointer to previous command.
11         ;
12 007614 010446 SLMVDN: MOV      R4, -(SP)
13         ;
14         ; Skip over nulls in front of current command
15         ;
16 007616 010504         MOV      R5, R4
17 007620 020427 000000G 1$:      CMP      R4, #SLLBUF      ; At top of buffer now?
18 007624 101002         BHI      2$                      ; Br if not
19 007626 012704 000000G         MOV      #SLEND, R4      ; Wrap around to the bottom
20 007632 105744 2$:      TSTB     -(R4)                  ; Is this character null?
21 007634 001003         BNE      3$                      ; Br if not
22 007636 020405         CMP      R4, R5                  ; Wrapped around to beginning?
23 007640 001367         BNE      1$                      ; Loop if not
24 007642 000416         BR       9$                      ; Nothing to recall
25         ;
26         ; Now skip back to null at front of the prev command
27         ;
28 007644 020427 000000G 3$:      CMP      R4, #SLLBUF      ; At top of buffer now?
29 007650 101002         BHI      4$                      ; Br if not
30 007652 012704 000000G         MOV      #SLEND, R4      ; Wrap around to the bottom
31 007656 105744 4$:      TSTB     -(R4)                  ; Is this character null?
32 007660 001371         BNE      3$                      ; Loop if not
33         ;
34         ; Now move forward to the 1st character of next command
35         ;
36 007662 005204 5$:      INC      R4                      ; Point to next char
37 007664 020427 000000G         CMP      R4, #SLEND      ; Past end of buffer?
38 007670 103402         BLO      6$                      ; Br if not
39 007672 012704 000000G         MOV      #SLLBUF, R4     ; Wrap around to the top
40         ;
41         ; Finished
42         ;
43 007676 010405 6$:      MOV      R4, R5                  ; Return pointer in R5
44 007700 012604 9$:      MOV      (SP)+, R4
45 007702 000207         RETURN

```

RSTLIN -- Restore a full line

```

1          .SBTTL  RSTLIN -- Restore a full line
2          ;-----
3          ; Restore a full line.
4          ;
5          ; Inputs:
6          ; R5 = Pointer to asciz string to be restored.
7          ;
8 007704 010246 RSTLIN: MOV     R2,-(SP)
9 007706 010346      MOV     R3,-(SP)
10 007710 010446      MOV     R4,-(SP)
11 007712 010546      MOV     R5,-(SP)
12 007714 010502      MOV     R5,R2      ; Save pointer to line start
13 007716 010504      MOV     R5,R4      ; "
14          ;
15          ; See if there is anything to restore
16          ;
17 007720 105712      TSTB     (R2)      ; Do we have a saved command?
18 007722 001003      BNE      1$        ; Br if yes
19 007724 004737 011154' CALL    RNGBEL    ; Ring bell if not
20 007730 000441      BR       9$
21          ;
22          ; Count number of characters in saved line
23          ;
24 007732 005005 1$:     CLR     R5      ; Count # chars in line
25 007734 004737 010046' 5$:   CALL    RSTCHR    ; Get next char from line buffer
26 007740 001402      BEQ      6$        ; Br if yes
27 007742 005205      INC     R5      ; Count # chars in saved line
28 007744 000773      BR       5$        ; Keep searching for the end of the line
29          ;
30          ; See if saved line will fit
31          ;
32 007746 004737 010522' 6$:   CALL    MAXLEN    ; Determine max allowed line length
33 007752 020500      CMP     R5,R0      ; Will saved line fit?
34 007754 101403      BLOS     4$        ; Br if yes
35 007756 004737 011154' CALL    RNGBEL    ; Ring bell if not
36 007762 000424      BR       9$
37          ;
38          ; Move saved line to edit buffer
39          ;
40 007764 012703 000000G 4$:   MOV     #SLEBUF,R3    ; Point to edit buffer
41 007770 010402      MOV     R4,R2      ; Point to start of saved line
42 007772 004737 010046' 2$:   CALL    RSTCHR    ; Get next char to be restored
43 007776 110023      MOVB    R0,(R3)+    ; Move saved char to edit buffer
44 010000 001374      BNE      2$        ; Loop until null moved
45          ;
46          ; Display the line
47          ;
48 010002 012704 000000G      MOV     #SLEBUF,R4    ; Display entire line
49 010006 004737 010336'      CALL    PAINT
50          ;
51          ; See if we should activate due to field width being filled
52          ;
53 010012 016102 000000G      MOV     LAFSIZ(R1),R2 ; Was field width activation specified?
54 010016 001406      BEQ      9$        ; Br if not
55 010020 020502      CMP     R5,R2      ; Does saved line = field width?
56 010022 103404      BLO     9$        ; Br if not
57 010024 004737 007172'      CALL    SAVLIN    ; Save the complete line

```

RSTLIN -- Restore a full line

58	010030	004737	007346'	CALL	LINFIN	; Input line is complete
59						
60					Finished	
61						
62	010034	012605		9\$: MOV	(SP)+, R5	
63	010036	012604		MOV	(SP)+, R4	
64	010040	012603		MOV	(SP)+, R3	
65	010042	012602		MOV	(SP)+, R2	
66	010044	000207		RETURN		

RSTLIN -- Restore a full line

```

1
2      ; -----
3      ;   Get the next character from a saved line and handle buffer wraparound.
4      ;
5      ;   Inputs:
6      ;       R2 = Points to next character to get.
7      ;
8      ;   Outputs:
9      ;       R0 = Character we got.
10     ;       R2 = Pointer to next character to get.
11     ;       Condition codes set for value returned in R0.
12     RSTCHR:
13     ;
14     ;   Get the character we want to return
15     ;
16     MOV     (R2)+, R0      ; Get character to be returned
17     ;
18     ;   See if we are getting from within SLLBUF and need to worry about wrap.
19     ;
20     CMP     R2, #SLEND    ; Are we at end of SLLBUF?
21     BNE     9$            ; Br if not
22     MOV     #SLLBUF, R2   ; Wrap around to top of SLLBUF
23     ;
24     ;   Finished
25     ;
26     9$:     TST     R0      ; Set sign for value in R0
27     RETURN

```

1				. SBTTL OSTRIK -- Overstrike
2				-----
3				; Replace the character under the cursor with a new (overstriking) character.
4				
5				; Inputs:
6				; R0 = New (overstriking) character.
7				; R1 = Job index number.
8				
9	010066	010246		OSTRIK: MOV R2, -(SP)
10	010070	010446		MOV R4, -(SP)
11				
12				; If we are positioned to the end of the line, do an insert
13				
14	010072	013704	000000G	MOV SLCX, R4 ;Get pointer to char under cursor
15	010076	105714		TSTB (R4) ;At end of line now?
16	010100	001006		BNE 1\$;Br if not
17	010102	012702	000000G	MOV #SLCBUF, R2 ;Point to char buffer
18	010106	110012		MOVB R0, (R2) ;Store char into insert buffer
19	010110	004737	010140'	CALL INSERT ;Insert the character
20	010114	000406		BR 9\$
21				
22				; We are positioned in the middle of the line.
23				; Replace the character being overstruck.
24				
25	010116	110014		1\$: MOVB R0, (R4) ;Replace char being overstruck
26	010120	004737	010336'	CALL PAINT ;Redisplay the line
27				
28				; Reposition the cursor
29				
30	010124	005204		INC R4 ;Position past overstrike
31	010126	004737	010642'	CALL CHRPOS ;Reposition cursor
32				
33				; Finished
34				
35	010132	012604		9\$: MOV (SP)+, R4
36	010134	012602		MOV (SP)+, R2
37	010136	000207		RETURN

INSERT -- Insert string into edit buffer

```

1          .SBTTL  INSERT -- Insert string into edit buffer
2          ;-----
3          ; INSERT is called to insert an asciz text string into the edit string.
4          ; The text is inserted immediately to the left of the cursor position.
5          ; After the insertion is made, the line is redisplayed and the cursor
6          ; is left pointing to the right of the inserted string.
7          ;
8          ; Inputs:
9          ; R1 = Job index number.
10         ; R2 = Pointer to asciz string to be inserted.
11         ;
12 010140 010246 INSERT: MOV      R2,-(SP)
13 010142 010346      MOV      R3,-(SP)
14 010144 010446      MOV      R4,-(SP)
15 010146 010546      MOV      R5,-(SP)
16         ;
17         ; Count number of characters in string being inserted
18         ;
19 010150 010203      MOV      R2,R3          ;Get pointer to insert string
20 010152 105723 1$:   TSTB     (R3)+        ;Search for null at end of string
21 010154 001376      BNE      1$           ;Loop till null hit
22 010156 160203      SUB      R2,R3        ;Get length of string
23 010160 005303      DEC      R3           ;Don't count null at end
24 010162 003003      BGT      7$           ;Br if there are some characters to insert
25         ;
26         ; There are no characters in string being inserted
27         ;
28 010164 004737 011154' CALL     RNGBEL      ;Ring the bell
29 010170 000455      BR       9$           ;We are finished
30         ;
31         ; Locate the end of the current edit line
32         ;
33 010172 013705 7$:   MOV      SLCX,R5      ;Get pointer to character under cursor
34 010176 010504      MOV      R5,R4
35 010200 105724 2$:   TSTB     (R4)+        ;Search for null at end of string
36 010202 001376      BNE      2$
37         ;
38         ; See if there is enough free space remaining in edit buffer to do
39         ; the insertion.
40         ;
41 010204 010405      MOV      R4,R5          ;Get pointer past null
42 010206 162705 000001G SUB      #SLEBUF+1,R5 ;Calc number of characters in current string
43 010212 060305      ADD      R3,R5        ;Add # characters in insert string
44 010214 004737 010522' CALL     MAXLEN      ;Compute maximum allowed line length
45 010220 020500      CMP      R5,R0        ;Compare line with insert with max allowed
46 010222 101403      BLOS     3$           ;Br if ok
47         ;
48         ; There is not enough space for the insertion
49         ;
50 010224 004737 011154' CALL     RNGBEL      ;Ring bell
51 010230 000435      BR       9$
52         ;
53         ; There is enough space to do the insertion.
54         ; Move characters over to make room for the insertion.
55         ;
56 010232 010405 3$:   MOV      R4,R5          ;Pointer past null at end of current string
57 010234 060305      ADD      R3,R5        ;Add number of characters to be inserted

```

INSERT -- Insert string into edit buffer

```

58 010236 114445          4$:      MOV      -(R4),-(R5)      ;Move characters to right
59 010240 020437 000000G      CMP      R4,SLCX          ;Have we moved enough?
60 010244 101374          BHI      4$                  ;Br if not
61                               ;
62                               ;   Insert the string
63                               ;
64 010246 010405          5$:      MOV      R4,R5          ;Get pointer to insertion point
65 010250 112225          6$:      MOV      (R2)+,(R5)+      ;Insert the string
66 010252 077302          SOB      R3,6$
67                               ;
68                               ;   Redisplay the line
69                               ;
70 010254 004737 010336'      CALL      PAINT            ;Redisplay the edit line
71                               ;
72                               ;   Reposition the cursor
73                               ;
74 010260 010504          MOV      R5,R4          ;Get desired cursor position
75 010262 004737 010642'      CALL      CHRPOS          ;Reposition cursor
76                               ;
77                               ;   Count total number of characters in the line and see if we should
78                               ;   activate due to field-width activation.
79                               ;
80 010266 016102 000000G      MOV      LAFSIZ(R1),R2      ;Is field-width activation in effect?
81 010272 001414          BEQ      9$                  ;Br if not
82 010274 013703 000000G      MOV      SLCX,R3          ;Point to cursor position
83 010300 105723          8$:      TSTB      (R3)+          ;Is there another character in buffer?
84 010302 001376          BNE      8$                  ;Loop till null hit
85 010304 162703 000001G      SUB      #SLEBUF+1,R3      ;Calc total number of characters in string
86 010310 020302          CMP      R3,R2          ;Have we reached activation size yet?
87 010312 103404          BLO      9$                  ;Br if not
88 010314 004737 007172'      CALL      SAVLIN          ;Save input line
89 010320 004737 007346'      CALL      LINFIN          ;Completed input line
90                               ;
91                               ;   Finished
92                               ;
93 010324 012605          9$:      MOV      (SP)+,R5
94 010326 012604          MOV      (SP)+,R4
95 010330 012603          MOV      (SP)+,R3
96 010332 012602          MOV      (SP)+,R2
97 010334 000207          RETURN

```

```

1          .SBTTL  PAINT  -- Redisplay current edit line
2          -----
3          ; PAINT is called to redisplay all or a portion of the current edit line.
4          ; The cursor is left pointing beyond the right end of the line.
5          ;
6          ; Inputs:
7          ;   R1 = Job index number.
8          ;   R4 = Pointer to character where redisplay is to begin.
9          ;
10         ; Outputs:
11         ;   SLCX  = Pointer to null at end of edit string.
12         ;   SLCCOL = Cursor position at end of line.
13         ;
14 010336 010246 PAINT:  MOV     R2,-(SP)
15 010340 010346      MOV     R3,-(SP)
16 010342 010446      MOV     R4,-(SP)
17 010344 010546      MOV     R5,-(SP)
18         ;
19         ; Position cursor to point where display is to begin
20         ;
21 010346 004737 010564'      CALL    COLCLC      ; Calculate position where redisplay begins
22 010352 004737 010664'      CALL    CURPOS      ; Position cursor there (R3 has column number)
23         ;
24         ; Begin loop to display each character.
25         ;
26 010356 112400 1$:        MOV     (R4)+,R0      ; Get next character to display
27 010360 001427      BEQ     5$                  ; Br if hit end of string
28 010362 120027 000000G      CMPB    R0,#TAB      ; Is this a tab?
29 010366 001015      BNE     3$                  ; Br if not tab
30         ;
31         ; Character being displayed is a tab.
32         ; Convert to spaces.
33         ;
34 010370 010300      MOV     R3,R0              ; Save position at start of tab field
35 010372 062703 000010      ADD     #8,R3        ; Calculate position beyond end of tab field
36 010376 042703 000007      BIC     #7,R3
37 010402 010302      MOV     R3,R2
38 010404 160002      SUB     R0,R2              ; Get # spaces needed to simulate the tab
39 010406 112700 000040      MOV     #' ,R0      ; Get a space character
40 010412 004737 011210' 2$:  CALL    ECHO        ; Send the space
41 010416 077203      SOB     R2,2$              ; Send as many as needed for tab
42 010420 000756      BR      1$
43         ;
44         ; Character is not a tab
45         ;
46 010422 004737 011210' 3$:  CALL    ECHO        ; Send the character to the terminal
47 010426 120027 000037      CMPB    R0,#37      ; Is this a printing character?
48 010432 101751      BLOS    1$                  ; Br if not
49 010434 005203      INC     R3                  ; Advance column number
50 010436 000747      BR      1$
51         ;
52         ; We finished displaying all characters in buffer.
53         ; See if we need to display blanks to wipe out deleted characters at end.
54         ;
55 010440 005304 5$:        DEC     R4              ; Point back to null at end of line
56 010442 010437 000000G      MOV     R4,SLCX      ; Save index where we want to leave cursor
57 010446 013702 000000G      MOV     SLECOL,R2    ; Get old end-of-line column number

```

PAINT -- Redisplay current edit line

```

58 010452 160302          SUB      R3,R2          ;Do we need to erase any old chars?
59 010454 003411          BLE      7$            ;Br if not
60                          ;
61                          ;   Display blanks to erase any characters that used to be at end of line
62                          ;
63 010456 010204          MOV      R2,R4          ;Save number of columns being erased
64 010460 116100 000000G  MOVB     LRBFIL(R1),R0    ;Get rubout-filler character for this line
65 010464 004737 011210' 6$:      CALL     ECHO          ;Send the space
66 010470 077203          SOB      R2,6$          ;Loop for as many as needed
67                          ;
68                          ;   Now leave cursor positioned at end of line
69                          ;
70 010472 004737 010776' 8$:      CALL     CSRLFT        ;Move cursor 1 column left
71 010476 077403          SOB      R4,8$          ;Move back over erased characters
72                          ;
73                          ;   Save information about column positions.
74                          ;
75 010500 010337 000000G  7$:      MOV      R3,SLCCOL      ;Cursor position
76 010504 010337 000000G      MOV      R3,SLECOL      ;End-of-line column
77                          ;
78                          ;   Finished
79                          ;
80 010510 012605          9$:      MOV      (SP)+,R5
81 010512 012604          MOV      (SP)+,R4
82 010514 012603          MOV      (SP)+,R3
83 010516 012602          MOV      (SP)+,R2
84 010520 000207          RETURN

```

```

1          .SBTTL  MAXLEN -- Compute maximum allowed line length
2          -----
3          ; MAXLEN is called to compute the maximum length allowed for an
4          ; edit line.  It takes into consideration the following constraints:
5          ; 1. Size of edit line buffer.
6          ; 2. Field width limit size.
7          ; 3. Field with activation size.
8          ;
9          ; Inputs:
10         ; R1 = Job index number.
11         ;
12         ; Outputs:
13         ; R0 = Maximum line length.
14         ;
15 010522 010246 MAXLEN: MOV      R2,-(SP)
16         ;
17         ; Start with edit buffer length.
18         ;
19 010524 012700 0000000 MOV      #SLMXLN,R0      ;Length of edit buffer
20         ;
21         ; See if a field width limit is in effect.
22         ;
23 010530 016102 0000000 MOV      LFWLIM(R1),R2    ;Is a field width limit in effect?
24 010534 001403 BEQ      1$                      ;Br if not
25 010536 020002 CMP      R0,R2                    ;Is current limit smaller?
26 010540 101401 BLOS     1$                      ;Br if yes
27 010542 010200 MOV      R2,R0                    ;Use field width limit
28         ;
29         ; See if field width activation is in effect.
30         ;
31 010544 016102 0000000 1$: MOV     LAFSIZ(R1),R2    ;Is field width activation in effect?
32 010550 001403 BEQ      2$                      ;Br if not
33 010552 020002 CMP      R0,R2                    ;Is current limit smaller?
34 010554 101401 BLOS     2$                      ;Br if yes
35 010556 010200 MOV      R2,R0                    ;Use field activation width as limit
36         ;
37         ; Finished
38         ;
39 010560 012602 2$: MOV     (SP)+,R2
40 010562 000207 RETURN

```

1				.SBTTL COLCLC -- Calculate column position of a character
2				-----
3				; COLCLC is called to compute the display column position of a specified
4				; character in the edit buffer.
5				;
6				; Inputs:
7				; R4 = Pointer to character whose position is to be computed.
8				;
9				; Outputs:
10				; R3 = Column position of the character.
11				;
12	010564	010246		COLCLC: MOV R2, -(SP)
13				;
14				; Initialize character info
15				;
16	010566	012702	000000G	MOV #SLEBUF, R2 ;Get pointer to start of edit buffer
17	010572	013703	000000G	MOV SLSCOL, R3 ;Get col # of left-most character
18				;
19				; Loop through all characters counting columns
20				;
21	010576	020204		1\$: CMP R2, R4 ;Have we reached character of interest?
22	010600	103016		BHIS 4\$;Br if yes
23				;
24				; Get next character to check
25				;
26	010602	112200		MOVB (R2)+, R0 ;Get next character
27	010604	120027	000000G	CMPB R0, #TAB ;Is character a tab?
28	010610	001005		BNE 2\$;Br if not
29				;
30				; Character is tab. Advance to next tab stop.
31				;
32	010612	062703	000010	ADD #8, R3 ;Advance to next tab stop
33	010616	042703	000007	BIC #7, R3
34	010622	000765		BR 1\$
35				;
36				; Character is not tab.
37				; See if it is a printing character.
38				;
39	010624	120027	000037	2\$: CMPB R0, #37 ;Is this a printing or control character?
40	010630	101762		BLOS 1\$;Br if control character
41	010632	005203		INC R3 ;Count one column for the character
42	010634	000760		BR 1\$;Go process more characters
43				;
44				; Finished
45				;
46	010636	012602		4\$: MOV (SP)+, R2
47	010640	000207		RETURN

CHRPOS -- Move cursor to a specific character

```

1          .SBTTL  CHRPOS -- Move cursor to a specific character
2          ;-----
3          ; CHRPOS is called to position the cursor to a specified character in the
4          ; edit buffer.
5          ;
6          ; Inputs:
7          ;   R4 = Pointer to character to which cursor is to be positioned.
8          ;
9 010642 010346 CHRPOS: MOV      R3, -(SP)
10         ;
11         ; Calculate position of desired character
12         ;
13 010644 004737 010564'      CALL    COLCLC      ; Calculate col position of character
14         ;
15         ; Position cursor there
16         ;
17 010650 004737 010664'      CALL    CURPOS      ; Position cursor
18         ;
19         ; Remember which character cursor is pointing to
20         ;
21 010654 010437 000000G      MOV      R4, SLCX      ; Save current cursor pointer
22         ;
23         ; Finished
24         ;
25 010660 012603      MOV      (SP)+, R3
26 010662 000207      RETURN

```

CURPOS -- Move cursor to a specified column

```

1          .SBTTL  CURPOS -- Move cursor to a specified column
2          ;-----
3          ; CURPOS is called to move the cursor to a specified column.
4          ;
5          ; Inputs:
6          ;   R3 = Position to which cursor is to be moved.
7          ;   SLCCOL = Current cursor position.
8          ;
9          ; Outputs:
10         ;   SLCCOL = Position where cursor has been positioned.
11         ;
12 010664 010246 CURPOS: MOV      R2,-(SP)
13         ;
14         ; Calculate how many columns cursor needs to be moved
15         ;
16 010666 010302         MOV      R3,R2          ;Get desired column position
17 010670 163702 000000G SUB      SLCCOL,R2      ;Calculate number of columns to move
18 010674 001413         BEQ      9$             ;Br if no movement necessary
19 010676 002404         BLT      2$             ;Br if need to move left
20         ;
21         ; Move cursor right
22         ;
23 010700 004737 010730' 1$:      CALL     CSRRIT      ;Move cursor right 1 column
24 010704 077203         SOB      R2,1$          ;Loop if need to move more
25 010706 000404         BR       4$
26         ;
27         ; Move cursor left
28         ;
29 010710 005402         2$:      NEG      R2          ;Get positive column count
30 010712 004737 010776' 3$:      CALL     CSRLFT      ;Move cursor left 1 column
31 010716 077203         SOB      R2,3$          ;Loop if need to move more
32         ;
33         ; Finished move
34         ;
35 010720 010337 000000G 4$:      MOV      R3,SLCCOL      ;Save new cursor position
36         ;
37         ; Finished
38         ;
39 010724 012602         9$:      MOV      (SP)+,R2
40 010726 000207         RETURN

```

CSRRIT -- Generate control sequence to move cursor right

```

1          .SBTTL  CSRRIT -- Generate control sequence to move cursor right
2          ;-----
3          ; CSRRIT is called to generate the terminal control sequence to move
4          ; the cursor right one column.
5          ;
6          ; Inputs:
7          ;   R1 = Job index number.
8          ;
9 010730 010246 CSRRIT: MOV      R2, -(SP)
10         ;
11         ; Get control sequence based on terminal type
12         ;
13 010732 012702 010766'      MOV      #MRS100, R2      ; Assume VT100 terminal
14 010736 004737 011602'      CALL     CHK52           ; Is this a VT52?
15 010742 103002              BCC      1$              ; Br if not
16 010744 012702 010772'      MOV      #MRS52, R2      ; Get VT52 control sequence
17         ;
18         ; Send control sequence to terminal
19         ;
20 010750 112200      1$:      MOVVB    (R2)+, R0        ; Get next character from sequence
21 010752 001403      BEQ      9$                      ; Br if hit end
22 010754 004737 011210'      CALL     ECHO            ; Send to terminal
23 010760 000773      BR       1$                      ; Continue sending
24         ;
25         ; Finished
26         ;
27 010762 012602      9$:      MOV      (SP)+, R2
28 010764 000207      RETURN
29         ;
30         ; Control sequences
31         ;
32 010766      0006      133      103 MRS100: .BYTE    ESC, '[, 'C, 0      ; VT100
33         010771      000
34 010772      0006      103      000 MRS52:  .BYTE    ESC, 'C, 0
                                     .EVEN

```

CSRLFT -- Generate control sequence to move cursor left

```
1          .SBTTL  CSRLFT -- Generate control sequence to move cursor left
2          ;-----
3          ; CSRLFT is called to generate the terminal control sequence to move
4          ; the cursor left one column.
5          ;
6          ; Inputs:
7          ; R1 = Job index number.
8          ;
9 010776 112700 000000G CSRLFT: MOVB  #BKSPAC,R0      ;Use backspace character to move to left
10 011002 004737 011210' CALL  ECHO
11          ;
12          ; Finished
13          ;
14 011006 000207          RETURN
```

CHKDLM -- Check for word delimiters

```

1          .SBTTL  CHKDLM -- Check for word delimiters
2          ;-----
3          ; Check to see if a character is a word delimiter.
4          ;
5          ; Inputs:
6          ;   RO = Character to check.
7          ;
8          ; Outputs:
9          ;   C-flag set      ==> Character is a delimiter.
10         ;   C-flag cleared ==> Character is not a delimiter.
11         ;
12 011010 010146  CHKDLM: MOV      R1,-(SP)
13 011012 010246      MOV      R2,-(SP)
14         ;
15         ; Null is always a delimiter
16         ;
17 011014 105700      TSTB      RO          ;Is character null?
18 011016 001407      BEQ       2$          ;Br if yes -- this is a delimiter
19         ;
20         ; Search table of delimiters for the character
21         ;
22 011020 012701 011052'  MOV      #WRDDLM,R1      ;Point to table of delimiters
23 011024 112102 1$:      MOVB     (R1)+,R2      ;Get next delimiter character
24 011026 001405      BEQ       3$          ;Br if hit end of table
25 011030 120002      CMPB      RO,R2        ;Is this character the delimiter?
26 011032 001401      BEQ       2$          ;Br if yes
27 011034 000773      BR        1$          ;Keep checking
28         ;
29         ; Character is a delimiter
30         ;
31 011036 000261 2$:      SEC          ;Signal that this is a delimiter
32 011040 000401      BR        9$
33         ;
34         ; Character is not a delimiter
35         ;
36 011042 000241 3$:      CLC          ;Signal that this is not a delimiter
37         ;
38         ; Finished
39         ;
40 011044 012602 9$:      MOV      (SP)+,R2
41 011046 012601      MOV      (SP)+,R1
42 011050 000207      RETURN
43         ;
44         ; Table of word delimiters
45         ;
46 011052 040 0006 054  WRDDLM: .BYTE  ' ',TAB,'.',',','=','/','\','.',',','[','(',')',0 ;Word delimiters (null=end)
47         ;
48         ;
49         ;
50         ;
51         ;
52         ;
53         ;
54         ;
55         ;
56         ;
57         ;
58         ;
59         ;
60         ;
61         ;
62         ;
63         ;
64         ;
65         ;
66         ;
67         ;
68         ;
69         ;
70         ;
71         ;
72         ;
73         ;
74         ;
75         ;
76         ;
77         ;
78         ;
79         ;
80         ;
81         ;
82         ;
83         ;
84         ;
85         ;
86         ;
87         ;
88         ;
89         ;
90         ;
91         ;
92         ;
93         ;
94         ;
95         ;
96         ;
97         ;
98         ;
99         ;
100        ;
101        ;
102        ;
103        ;
104        ;
105        ;
106        ;
107        ;
108        ;
109        ;
110        ;
111        ;
112        ;
113        ;
114        ;
115        ;
116        ;
117        ;
118        ;
119        ;
120        ;
121        ;
122        ;
123        ;
124        ;
125        ;
126        ;
127        ;
128        ;
129        ;
130        ;
131        ;
132        ;
133        ;
134        ;
135        ;
136        ;
137        ;
138        ;
139        ;
140        ;
141        ;
142        ;
143        ;
144        ;
145        ;
146        ;
147        ;
148        ;
149        ;
150        ;
151        ;
152        ;
153        ;
154        ;
155        ;
156        ;
157        ;
158        ;
159        ;
160        ;
161        ;
162        ;
163        ;
164        ;
165        ;
166        ;
167        ;
168        ;
169        ;
170        ;
171        ;
172        ;
173        ;
174        ;
175        ;
176        ;
177        ;
178        ;
179        ;
180        ;
181        ;
182        ;
183        ;
184        ;
185        ;
186        ;
187        ;
188        ;
189        ;
190        ;
191        ;
192        ;
193        ;
194        ;
195        ;
196        ;
197        ;
198        ;
199        ;
200        ;
201        ;
202        ;
203        ;
204        ;
205        ;
206        ;
207        ;
208        ;
209        ;
210        ;
211        ;
212        ;
213        ;
214        ;
215        ;
216        ;
217        ;
218        ;
219        ;
220        ;
221        ;
222        ;
223        ;
224        ;
225        ;
226        ;
227        ;
228        ;
229        ;
230        ;
231        ;
232        ;
233        ;
234        ;
235        ;
236        ;
237        ;
238        ;
239        ;
240        ;
241        ;
242        ;
243        ;
244        ;
245        ;
246        ;
247        ;
248        ;
249        ;
250        ;
251        ;
252        ;
253        ;
254        ;
255        ;
256        ;
257        ;
258        ;
259        ;
260        ;
261        ;
262        ;
263        ;
264        ;
265        ;
266        ;
267        ;
268        ;
269        ;
270        ;
271        ;
272        ;
273        ;
274        ;
275        ;
276        ;
277        ;
278        ;
279        ;
280        ;
281        ;
282        ;
283        ;
284        ;
285        ;
286        ;
287        ;
288        ;
289        ;
290        ;
291        ;
292        ;
293        ;
294        ;
295        ;
296        ;
297        ;
298        ;
299        ;
300        ;
301        ;
302        ;
303        ;
304        ;
305        ;
306        ;
307        ;
308        ;
309        ;
310        ;
311        ;
312        ;
313        ;
314        ;
315        ;
316        ;
317        ;
318        ;
319        ;
320        ;
321        ;
322        ;
323        ;
324        ;
325        ;
326        ;
327        ;
328        ;
329        ;
330        ;
331        ;
332        ;
333        ;
334        ;
335        ;
336        ;
337        ;
338        ;
339        ;
340        ;
341        ;
342        ;
343        ;
344        ;
345        ;
346        ;
347        ;
348        ;
349        ;
350        ;
351        ;
352        ;
353        ;
354        ;
355        ;
356        ;
357        ;
358        ;
359        ;
360        ;
361        ;
362        ;
363        ;
364        ;
365        ;
366        ;
367        ;
368        ;
369        ;
370        ;
371        ;
372        ;
373        ;
374        ;
375        ;
376        ;
377        ;
378        ;
379        ;
380        ;
381        ;
382        ;
383        ;
384        ;
385        ;
386        ;
387        ;
388        ;
389        ;
390        ;
391        ;
392        ;
393        ;
394        ;
395        ;
396        ;
397        ;
398        ;
399        ;
400        ;
401        ;
402        ;
403        ;
404        ;
405        ;
406        ;
407        ;
408        ;
409        ;
410        ;
411        ;
412        ;
413        ;
414        ;
415        ;
416        ;
417        ;
418        ;
419        ;
420        ;
421        ;
422        ;
423        ;
424        ;
425        ;
426        ;
427        ;
428        ;
429        ;
430        ;
431        ;
432        ;
433        ;
434        ;
435        ;
436        ;
437        ;
438        ;
439        ;
440        ;
441        ;
442        ;
443        ;
444        ;
445        ;
446        ;
447        ;
448        ;
449        ;
450        ;
451        ;
452        ;
453        ;
454        ;
455        ;
456        ;
457        ;
458        ;
459        ;
460        ;
461        ;
462        ;
463        ;
464        ;
465        ;
466        ;
467        ;
468        ;
469        ;
470        ;
471        ;
472        ;
473        ;
474        ;
475        ;
476        ;
477        ;
478        ;
479        ;
480        ;
481        ;
482        ;
483        ;
484        ;
485        ;
486        ;
487        ;
488        ;
489        ;
490        ;
491        ;
492        ;
493        ;
494        ;
495        ;
496        ;
497        ;
498        ;
499        ;
500        ;
501        ;
502        ;
503        ;
504        ;
505        ;
506        ;
507        ;
508        ;
509        ;
510        ;
511        ;
512        ;
513        ;
514        ;
515        ;
516        ;
517        ;
518        ;
519        ;
520        ;
521        ;
522        ;
523        ;
524        ;
525        ;
526        ;
527        ;
528        ;
529        ;
530        ;
531        ;
532        ;
533        ;
534        ;
535        ;
536        ;
537        ;
538        ;
539        ;
540        ;
541        ;
542        ;
543        ;
544        ;
545        ;
546        ;
547        ;
548        ;
549        ;
550        ;
551        ;
552        ;
553        ;
554        ;
555        ;
556        ;
557        ;
558        ;
559        ;
560        ;
561        ;
562        ;
563        ;
564        ;
565        ;
566        ;
567        ;
568        ;
569        ;
570        ;
571        ;
572        ;
573        ;
574        ;
575        ;
576        ;
577        ;
578        ;
579        ;
580        ;
581        ;
582        ;
583        ;
584        ;
585        ;
586        ;
587        ;
588        ;
589        ;
590        ;
591        ;
592        ;
593        ;
594        ;
595        ;
596        ;
597        ;
598        ;
599        ;
600        ;
601        ;
602        ;
603        ;
604        ;
605        ;
606        ;
607        ;
608        ;
609        ;
610        ;
611        ;
612        ;
613        ;
614        ;
615        ;
616        ;
617        ;
618        ;
619        ;
620        ;
621        ;
622        ;
623        ;
624        ;
625        ;
626        ;
627        ;
628        ;
629        ;
630        ;
631        ;
632        ;
633        ;
634        ;
635        ;
636        ;
637        ;
638        ;
639        ;
640        ;
641        ;
642        ;
643        ;
644        ;
645        ;
646        ;
647        ;
648        ;
649        ;
650        ;
651        ;
652        ;
653        ;
654        ;
655        ;
656        ;
657        ;
658        ;
659        ;
660        ;
661        ;
662        ;
663        ;
664        ;
665        ;
666        ;
667        ;
668        ;
669        ;
670        ;
671        ;
672        ;
673        ;
674        ;
675        ;
676        ;
677        ;
678        ;
679        ;
680        ;
681        ;
682        ;
683        ;
684        ;
685        ;
686        ;
687        ;
688        ;
689        ;
690        ;
691        ;
692        ;
693        ;
694        ;
695        ;
696        ;
697        ;
698        ;
699        ;
700        ;
701        ;
702        ;
703        ;
704        ;
705        ;
706        ;
707        ;
708        ;
709        ;
710        ;
711        ;
712        ;
713        ;
714        ;
715        ;
716        ;
717        ;
718        ;
719        ;
720        ;
721        ;
722        ;
723        ;
724        ;
725        ;
726        ;
727        ;
728        ;
729        ;
730        ;
731        ;
732        ;
733        ;
734        ;
735        ;
736        ;
737        ;
738        ;
739        ;
740        ;
741        ;
742        ;
743        ;
744        ;
745        ;
746        ;
747        ;
748        ;
749        ;
750        ;
751        ;
752        ;
753        ;
754        ;
755        ;
756        ;
757        ;
758        ;
759        ;
760        ;
761        ;
762        ;
763        ;
764        ;
765        ;
766        ;
767        ;
768        ;
769        ;
770        ;
771        ;
772        ;
773        ;
774        ;
775        ;
776        ;
777        ;
778        ;
779        ;
780        ;
781        ;
782        ;
783        ;
784        ;
785        ;
786        ;
787        ;
788        ;
789        ;
790        ;
791        ;
792        ;
793        ;
794        ;
795        ;
796        ;
797        ;
798        ;
799        ;
800        ;
801        ;
802        ;
803        ;
804        ;
805        ;
806        ;
807        ;
808        ;
809        ;
810        ;
811        ;
812        ;
813        ;
814        ;
815        ;
816        ;
817        ;
818        ;
819        ;
820        ;
821        ;
822        ;
823        ;
824        ;
825        ;
826        ;
827        ;
828        ;
829        ;
830        ;
831        ;
832        ;
833        ;
834        ;
835        ;
836        ;
837        ;
838        ;
839        ;
840        ;
841        ;
842        ;
843        ;
844        ;
845        ;
846        ;
847        ;
848        ;
849        ;
850        ;
851        ;
852        ;
853        ;
854        ;
855        ;
856        ;
857        ;
858        ;
859        ;
860        ;
861        ;
862        ;
863        ;
864        ;
865        ;
866        ;
867        ;
868        ;
869        ;
870        ;
871        ;
872        ;
873        ;
874        ;
875        ;
876        ;
877        ;
878        ;
879        ;
880        ;
881        ;
882        ;
883        ;
884        ;
885        ;
886        ;
887        ;
888        ;
889        ;
890        ;
891        ;
892        ;
893        ;
894        ;
895        ;
896        ;
897        ;
898        ;
899        ;
900        ;
901        ;
902        ;
903        ;
904        ;
905        ;
906        ;
907        ;
908        ;
909        ;
910        ;
911        ;
912        ;
913        ;
914        ;
915        ;
916        ;
917        ;
918        ;
919        ;
920        ;
921        ;
922        ;
923        ;
924        ;
925        ;
926        ;
927        ;
928        ;
929        ;
930        ;
931        ;
932        ;
933        ;
934        ;
935        ;
936        ;
937        ;
938        ;
939        ;
940        ;
941        ;
942        ;
943        ;
944        ;
945        ;
946        ;
947        ;
948        ;
949        ;
950        ;
951        ;
952        ;
953        ;
954        ;
955        ;
956        ;
957        ;
958        ;
959        ;
960        ;
961        ;
962        ;
963        ;
964        ;
965        ;
966        ;
967        ;
968        ;
969        ;
970        ;
971        ;
972        ;
973        ;
974        ;
975        ;
976        ;
977        ;
978        ;
979        ;
980        ;
981        ;
982        ;
983        ;
984        ;
985        ;
986        ;
987        ;
988        ;
989        ;
990        ;
991        ;
992        ;
993        ;
994        ;
995        ;
996        ;
997        ;
998        ;
999        ;
1000       ;

```

CVTLC -- Convert lower-case characters to upper-case

```

1          .SBTTL  CVTLC  -- Convert lower-case characters to upper-case
2          ;-----
3          ; CVTLC is called to convert lower-case characters to upper case
4          ; if requested by the currently running program.
5          ;
6          ; Inputs:
7          ;   R0 = Input character.
8          ;   R1 = Job index number.
9          ;
10         ; Outputs:
11         ;   R0 = Converted character.
12         ;
13 011066  032761  000000G 000000G CVTLC:  BIT    #$LC,LSW2(R1)  ; Was "SET TT LC" done?
14 011074  001404                      BEQ     1$                ; Br if not
15 011076  032761  000000G 000000G      BIT    #LCBIT,LSW(R1) ; Does program want lower case characters?
16 011104  001010                      BNE     2$                ; Br if yes -- lower-case chars ok
17         ;
18         ; Translate lower-case character to upper-case
19         ;
20 011106  120027  000141      1$:      CMPB    R0,#141          ; Is this a lower case letter?
21 011112  103405                      BLO     2$                ; Br if not
22 011114  120027  000172      CMPB    R0,#172
23 011120  101002                      BHI     2$                ; Br if not lower-case letter
24 011122  042700  000040      BIC     #40,R0                    ; Convert lower-case to upper-case
25         ;
26         ; Finished
27         ;
28 011126  000207      2$:      RETURN

```

ECOCTL -- Echo a control character

```

1          .SBTTL  ECOCTL -- Echo a control character
2          -----
3          ; ECOCTL is called to echo a control character by printing "^x" where
4          ; "x" is the character.
5          ;
6          ; Inputs:
7          ;   R1 = Job index number.
8          ;   R5 = Control character.
9          ;
10         ECOCTL:
11         ;
12         ; Echo "^"
13         ;
14         MOVW    #'^,R0          ;Get up-arrow
15         CALL    ECHO            ;Echo it
16         ;
17         ; Echo character
18         ;
19         MOVW    R5,R0          ;Get control character
20         ADD     #100,R0        ;Convert to upper-case character
21         CALL    ECHO            ;Echo it
22         ;
23         ; Finished
24         ;
25         RETURN
26
27         .SBTTL  RINGBEL -- Ring bell to signal error
28         -----
29         ; RINGBEL is called to send a bell character to the terminal to signal
30         ; an error condition.
31         ;
32         RINGBEL: MOVW    #BELL,R0      ;Get bell character
33         CALL    ECHO2           ;Send to terminal
34         ;
35         ; Finished
36         ;
37         RETURN
38
39         .SBTTL  AKEYON -- Turn on alternate keypad mode
40         -----
41         ; AKEYON is called to enable alternate keypad mode.
42         ;
43         ; Inputs:
44         ;   R1 = Job index number.
45         ;
46         AKEYON:
47         ;
48         ; Send control sequence to terminal
49         ;
50         MOVW    #ESC,R0          ;Send Escape as 1st char
51         CALL    ECHO2           ;Send to terminal
52         MOVW    #'=,R0          ;Send equal sign as 2nd char
53         CALL    ECHO2
54         ;
55         ; Finished
56         ;
57         RETURN

```

```

1          .SBTTL  ECHO  -- Send character to the terminal
2          ;-----
3          ; ECHO is called to send a character to the terminal.
4          ; ECHO2 always echoes the character even if character echoing is turned off.
5          ;
6          ; Inputs:
7          ;   R1 = Job index number.
8          ;   R0 = Character to be sent (preserved).
9          ;
10         ECHO:
11         ;
12         ; See if character echoing is wanted.
13         ;
14         011210 032761 000000G 000000G      BIT    #$ECHO,LSW2(R1) ;Is character echoing wanted?
15         011216 001402                      BEQ    9$           ;Br if not
16         011220 004737 011226'              CALL   ECHO2        ;Echo the character
17         011224 000207                      9$:    RETURN
18
19         ;-----
20         ; We want character echoing
21         ;
22         011226 010046                      ECHO2:  MOV     R0,-(SP) ;Save character being echoed
23         ;
24         ; Send character to terminal
25         ;
26         011230                      OCALL   QUECHR        ;Send char to terminal
27         ;
28         ; Finished
29         ;
30         011236 012600                      MOV     (SP)+,R0    ;Recover character
31         011240 000207                      9$:    RETURN

```

KEYCK -- Check to see if key is defined

```

1          .SBTTL  KEYCK  -- Check to see if key is defined
2          ;-----
3          ; KEYCK is the routine called through the use of the KEYCHK macro.
4          ; It sets up information correctly in registers and calls KEYSRC.
5          ;
6          ; Form of the call:
7          ;     JSR      R5,KEYCK
8          ;     .WORD   key_code
9          ;
10         KEYCK:
11         ;
12         ; Pick up the key code.
13         ;
14         MOV      (R5)+,R0          ;Get key code
15         ;
16         ; See if gold key has been pressed
17         ;
18         TSTB     SLGOLD             ;Has gold key been pressed?
19         BEQ      1$                ;Br if not
20         BIS      #KT$GLD*400,R0    ;Set key type to gold
21         BR       2$
22         BIS      #KT$NRM*400,R0    ;Set normal key type
23         ;
24         ; See if this key is defined by user
25         ;
26         1$:      CALL     KEYSRC
27         ;
28         ; Finished
29         ;
30         RTS      R5                ;Return to caller

```

KEYSRC -- See if terminal key is defined by user

```

1          .SBTTL  KEYSRC -- See if terminal key is defined by user
2          -----
3          ; Determine if a specific terminal key is defined by the user.
4          ; If so, perform the key substitution.
5          ;
6          ; Inputs:
7          ;   R0 = Key code to be searched for.
8          ;   R1 = Job index number.
9          ;
10         ; Outputs:
11         ;   C-flag set      ==> Key was defined; processing has been done for it.
12         ;   C-flag cleared ==> Key was not defined.
13         ;
14 011272   KEYSRC:
15         ;
16         ; See if there are any user-defined keys
17         ;
18 011272   005737   000000G       TST      KEYPAR      ;Are there any user-defined keys?
19 011276   001404       BEQ      5$              ;Br if not
20 011300   032761   000000G 000000G  BIT      ##SLLET,LSW7(R1);Is substitution wanted?
21 011306   001002       BNE      4$              ;Br if yes
22 011310   000241   5$:      CLC                  ;Signal failure on return
23 011312   000207       RETURN
24         ;
25         ; There are some user-defined keys
26         ;
27 011314   010246   4$:      MOV      R2,-(SP)
28 011316   010346       MOV      R3,-(SP)
29         ;
30         ; Begin loop to search for the key definition
31         ;
32 011320   013703   000000G       MOV      VKEYMX,R3      ;Get max # key definitions allowed
33 011324   012702   000000G       MOV      #VPAR6,R2      ;Point to virtual address of region base
34 011330       KEYMAP      ;;;Map to key region
35 011352   020062   000000G   1$:  CMP      R0,KD$COD(R2)  ;;;Is this entry for the key?
36 011356   001413       BEQ      2$              ;;;Br if yes
37 011360   062702   000000G       ADD      #KD$$SZ,R2     ;;;Point to next entry
38 011364   077306       SOB      R3,1$          ;;;Loop if more to check
39 011366       KEYUMP      ;Restore par 6 mapping
40         ;
41         ; Key is not defined
42         ;
43 011402   000241   3$:      CLC                  ;Signal failure on return
44 011404   000415       BR       9$
45         ;
46         ; We found the key definition
47         ;
48 011406   116203   000000G   2$:  MOVB     KD$FLG(R2),R3  ;;;Get control flags for key def
49 011412       KEYUMP      ;Restore par 6 mapping
50 011426   105037   000000G       CLRB     SLGOLD        ;Clear gold key flag
51         ;
52         ; Call routine which substitutes the key definition string
53         ;
54 011432   004737   011446'       CALL     KEYSUB        ;Do the substitution
55 011436   000261       SEC                  ;Set flag saying we did a key definition
56         ;
57         ; Finished

```

TSSLE -- TSX-Plus Single Line E MACRO V05.05 Wednesday 18-Jan-89 14:16 Page 93-1
KEYSRC -- See if terminal key is defined by user

58				
59	011440	012603	9\$:	MOV (SP)+, R3
60	011442	012602		MOV (SP)+, R2
61	011444	000207		RETURN

KEYSUB -- Substitute a defined string for a key

```

1          .SBTTL  KEYSUB -- Substitute a defined string for a key
2          ;-----
3          ; A user-defined key has been identified.
4          ; Insert the defined text into the command buffer.
5          ;
6          ; Inputs:
7          ;   R1 = Current job index number
8          ;   R2 = Pointer to key definition block
9          ;   R3 = Control flags from definition block
10         ;
11 011446 010246 KEYSUB: MOV      R2,-(SP)
12 011450 010546      MOV      R5,-(SP)
13 011452 016146 000000G      MOV      LSW2(R1),-(SP) ; Save current LSW2 flag settings
14         ;
15         ; See if we should suppress display of this string
16         ;
17 011456 132703 000000G      BITB     #KF$ECHO,R3      ; Should we echo the string?
18 011462 001003      BNE      1$      ; Br if yes
19 011464 042761 000000G 000000G      BIC      #$ECHO,LSW2(R1) ; Suppress echo of string
20         ;
21         ; Move the characters from the string definition to input line
22         ;
23 011472 062702 000000G 1$:      ADD      #KD$TXT,R2      ; Point to key def string
24         ;
25         ; Get next character from definition string
26         ;
27 011476      2$:      KEYMAP      ;;; Map to definition buffer
28 011520 112205      MOVB      (R2)+,R5      ;;; Get next char from definition string
29 011522      KEYUMP      ; Restore par 6 mapping
30 011536 105705      TSTB      R5      ; Did we reach the end of the string?
31 011540 001403      BEQ      5$      ; Br if yes
32         ;
33         ; Process the character as an ordinary received character
34         ;
35 011542 004737 000450'      CALL      ORDCHR      ; Process the character
36 011546 000753      BR      2$      ; Go see if there are more characters
37         ;
38         ; We reached the end of the string.
39         ; See if we should terminate the input line.
40         ;
41 011550 132703 000000G 5$:      BITB     #KF$TRM,R3      ; Should we terminate the input line?
42 011554 001405      BEQ      9$      ; Br if not
43         ;
44         ; Terminate the current input line
45         ;
46 011556 112737 000001 000000G      MOVB      #1,SLCR      ; Say CR-LF received
47 011564 004737 005622'      CALL      ICPCR      ; Pretend we received a carriage-return
48         ;
49         ; Finished
50         ;
51 011570 012661 000000G 9$:      MOV      (SP)+,LSW2(R1) ; Restore echo flag
52 011574 012605      MOV      (SP)+,R5
53 011576 012602      MOV      (SP)+,R2
54 011600 000207      RETURN

```

```

1          .SBTTL  CHK52  -- Determine if terminal is a VT52
2          ;-----
3          ; Determine if the terminal is a VT52.
4          ;
5          ; Inputs:
6          ;   R1 = Job index number
7          ;
8          ; Outputs:
9          ;   C-flag set if this is a VT52
10         ;   All registers are preserved.
11         ;
12 011602 010246  CHK52:  MOV      R2,-(SP)
13         ;
14         ; See if the terminal type is VT52
15         ;
16 011604 032761 000000G 000000G      BIT      #VT52,LTRMTP(R1);Is the terminal type VT52?
17 011612 001010      BNE      5$      ;Br if yes
18         ;
19         ; See if we are emulating a VT52
20         ;
21 011614 116102 000000G      MOVB      LNPRIM(R1),R2 ;Get primary job index number
22 011620 032762 000000G 000000G      BIT      #$V52EM,LSW11(R2);Are we emulating a VT52?
23 011626 001002      BNE      5$      ;Br if yes
24         ;
25         ; This is not a VT52
26         ;
27 011630 000241      CLC                      ;Signal not VT52
28 011632 000401      BR      9$
29         ;
30         ; This is a VT52
31         ;
32 011634 000261 5$:      SEC                      ;Signal that this is a VT52
33         ;
34         ; Finished
35         ;
36 011636 012602 9$:      MOV      (SP)+,R2
37 011640 000207      RETURN
38         .END
  
```

Errors detected: 0

*** Assembler statistics

Work file reads: 0
 Work file writes: 0
 Size of work file: 8356 Words (33 Pages)
 Size of core pool: 18176 Words (71 Pages)
 Operating system: RT-11

Elapsed time: 00:01:43.49
 ,LP:TSSLE=DK:TSSLE/C/N:SYM

\$1ESC	1-24	59-34											
\$CTRLW	1-24	7-16	7-18	59-30	59-32	59-35							
\$DBGMD	1-39	6-38											
\$DETC	1-45	3-17											
\$DISCN	1-39	3-19											
\$ECHO	1-41	91-14	94-19										
\$LC	1-40	89-13											
\$NDICP	1-43	71-14											
\$NDIN	1-36	71-20											
\$NOINT	1-43	2-51	71-26										
\$NOLF	1-40	60-51											
\$PWKEY	1-23	59-16											
\$SCCA	1-22	65-16											
\$SLINI	1-40	2-16	4-62	65-41									
\$SLKED	1-41	4-13											
\$SLLET	1-25	93-20											
\$SUCF	1-42	71-21											
\$V52EM	1-23	95-22											
\$VBELL	1-23	7-24											
\$WDISP	1-24	7-25											
\$XSTOP	1-35	71-13											
AKEYON	4-15	90-46#											
BELL	1-38	90-32											
BKSPAC	1-35	87-9											
CHK52	20-17	26-22	29-17	32-17	63-15	86-14	95-12#						
CHKABT	1-36	71-22											
CHKDLM	24-29	24-36	24-43	24-58	24-66	34-26	34-30	34-37	61-44	61-62	88-12#		
CHRPOS	16-20	16-34	17-23	17-37	21-50	23-23	24-75	24-107	25-29	25-35	26-43	33-24	
	33-38	34-64	34-76	61-89	62-30	62-44	66-17	67-54	68-17	74-37	79-31	80-75	
	84-9#												
CKRDTM	6-48	9-13#											
CKUAC	6-43	8-14#											
COLCLC	81-21	83-12#	84-13										
CR	1-37	60-43	60-49	68-25									
CSCHK	5-14	10-14#											
CSEND	12-36	13-66#											
CSFIN	11-42	11-70	12-10#										
CSICHR	1-44	10-25											
CSRLFT	81-70	85-30	87-9#										
CSRRIT	85-23	86-9#											
CSTBL	12-18	13-10#											
CTLRTN	59-41	59-51#											
CTRLQ	1-33												
CURPOS	81-22	84-17	85-12#										
CVTLC	7-33	8-26	89-13#										
DELCHR	1-39	3-37											
DELLFT	33-39	69-28	74-13#										
DOCTRL	6-55	59-10#											
DOSWIT	1-24	7-23											
E1	13-33	37-8#											
E2	13-34	38-8#											
E3	13-35	39-8#											
E4	13-36	40-8#											
E5	13-37	41-8#											
E6	13-38	42-8#											
ECHO	60-50	60-54	81-40	81-46	81-65	86-22	87-10	90-15	90-21	91-10#			

ECHO2	68-26	68-28	90-33	90-51	90-53	91-16	91-22#
ECDOCTL	65-29	68-21	90-10#				
ENQTL	1-42	73-17					
EP52	11-22#	63-17					
EPCH1	11-12#	63-14					
EPCH2	11-15	11-35	11-51#	63-32	63-45		
ESC	1-40	10-18	86-32	86-33	90-50		
F10	13-43	47-8#					
F11	13-44	48-8#					
F12	13-45	49-8#					
F13	13-46	50-8#					
F14	13-47	51-8#					
F15	13-48	52-8#					
F16	13-49	53-8#					
F17	13-50	54-8#					
F18	13-51	55-8#					
F19	13-52	56-8#					
F20	13-53	57-8#					
F6	13-39	43-8#					
F7	13-40	44-8#					
F8	13-41	45-8#					
F9	13-42	46-8#					
GTSLCH	1-18	2-12#					
ICPBS	49-24	59-59	62-8#				
ICPCR	22-16	59-64	60-9#	61-15	94-47		
ICPCSI	10-27	63-32#					
ICPCTA	59-52	64-5#					
ICPCTC	59-54	65-9#					
ICPCTR	59-69	59-74	66-5#				
ICPCTU	59-72	67-9#					
ICPCTZ	59-77	65-15	68-9#				
ICPESC	10-20	59-78	63-9#				
ICPLF	50-24	59-61	61-9#				
ICPNUL	59-51	70-5#					
ICPRUB	6-59	69-5#					
ICPSS3	10-34	63-45#					
INSERT	7-43	21-49	33-21	34-75	61-32	67-21	69-15
INTPRI	1-34	71-36	93-39	93-49	94-29	79-19	80-12#
INWAIT	3-30	71-8#					
KBS	6-23	35-5#					
KBX	6-33	36-5#					
KC\$COM	1-30	33-10					
KC\$DOT	1-30	58-9					
KC\$DWN	1-32	15-10					
KC\$E1	1-27	37-12					
KC\$E2	1-27	38-12					
KC\$E3	1-27	39-12					
KC\$E4	1-27	40-12					
KC\$E5	1-27	41-12					
KC\$E6	1-27	42-12					
KC\$ENT	1-30	22-9					
KC\$F10	1-28	47-12					
KC\$F11	1-28	48-12					
KC\$F12	1-28	49-12					
KC\$F13	1-28	50-12					
KC\$F14	1-29	51-12					

[illegible]

TCINVL	22-23#				
TCLEFT	13-14	13-57	13-61	16-8#	28-24
TCPF1	13-16	13-62	18-8#		
TCPF2	13-17	13-63	19-8#		
TCPF3	13-18	13-64	20-8#		
TCPF4	13-19	13-65	20-18	21-8#	25-41
TCRITE	13-13	13-56	13-60	17-8#	27-24
TCUP	13-11	13-54	13-58	14-8#	
TSSLE	1-6#				
VINTIO	1-37	2-53	71-28		
VKEYMX	1-26	93-32			
VPAR6	1-32	93-33			
VQUAN1	1-42	2-54	71-29		
VT100	1-42				
VT52	1-35	95-16			
VVLSCH	1-24	59-28			
VVPWCH	1-25	59-14			
WINDSP	1-24	7-27			
WINPRT	1-25	59-23			
WRDDL	88-22	88-46#			

Cross reference table (CREF V05.05)

[illegible]