```
TTTTTTTTTTTTTTTT  TTTTTTTTTTTTTTTT  DDDDDDDDDDDD     RRRRRRRRRRRR     VVV         VVV  RRRRRRRRRRRR
TTTTTTTTTTTTTTTT  TTTTTTTTTTTTTTTT  DDDDDDDDDDDDD    RRRRRRRRRRRRR    VVV         VVV  RRRRRRRRRRRRR
TTTTTTTTTTTTTTTT  TTTTTTTTTTTTTTTT  DDDDDDDDDDDDD    RRRRRRRRRRRRR    VVV         VVV  RRRRRRRRRRRRR
      TTT               TTT         DDD          DDD  RRR          RRR  VVV         VVV  RRR          RRR
      TTT               TTT         DDD          DDD  RRR          RRR  VVV         VVV  RRR          RRR
      TTT               TTT         DDD          DDD  RRR          RRR  VVV         VVV  RRR          RRR
      TTT               TTT         DDD          DDD  RRR          RRR  VVV         VVV  RRR          RRR
      TTT               TTT         DDD          DDD  RRR          RRR  VVV         VVV  RRR          RRR
      TTT               TTT         DDD          DDD  RRR          RRR  VVV         VVV  RRR          RRR
      TTT               TTT         DDD          DDD  RRRRRRRRRRRR     VVV         VVV  RRRRRRRRRRRR
      TTT               TTT         DDD          DDD  RRRRRRRRRRRR     VVV         VVV  RRRRRRRRRRRR
      TTT               TTT         DDD          DDD  RRRRRRRRRRRR     VVV         VVV  RRRRRRRRRRRR
      TTT               TTT         DDD          DDD  RRR   RRR         VVV         VVV  RRR   RRR
      TTT               TTT         DDD          DDD  RRR   RRR         VVV         VVV  RRR   RRR
      TTT               TTT         DDD          DDD  RRR   RRR         VVV         VVV  RRR   RRR
      TTT               TTT         DDD          DDD  RRR      RRR        VVV   VVV     RRR      RRR
      TTT               TTT         DDD          DDD  RRR      RRR        VVV   VVV     RRR      RRR
      TTT               TTT         DDD          DDD  RRR      RRR        VVV   VVV     RRR      RRR
      TTT               TTT         DDDDDDDDDDDDD     RRR         RRR        VVV         RRR         RRR
      TTT               TTT         DDDDDDDDDDDDD     RRR         RRR        VVV         RRR         RRR
      TTT               TTT         DDDDDDDDDDDDD     RRR         RRR        VVV         RRR         RRR
```

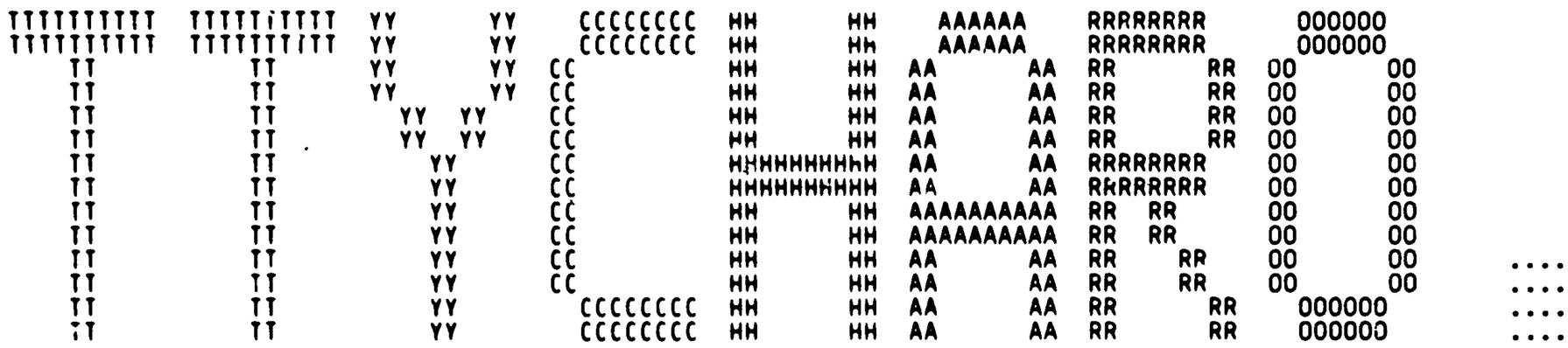
TTTTTTTTTT TTTTTTTTTT TT TT TT TT TT TT TT TT TT TT TT TT TT TT
YY YY YY YY YY YY YY YY YY YY YY YY YY YY
CCCCCCCC CCCCCCCC CC CC CC CC CC CC CC CC CCCCCCCC CCCCCCCC
HH HH HH HH HH HH HH HH HH HH HH HH HH HH
AAAAAA AAAAAA AA AA AA AA AA AA AA AA AA AA AA AA
RRRRRRRR RRRRRRRR RR RR RR RR RRRRRRRR RRRRRRRR RR RR RR RR
000000 000000 00 00 00 00 00 00 00 00 00 00 00 00
LL LL LL LL LL LL LL LL LL LL LL LLLLLLLLLL LLLLLLLLLL
IIIIII IIIIII II II II II II II II II IIIIII IIIIII
SSSSSSSS SSSSSSSS SS SS SS SS SSSSSS SSSSSS SS SS SS SS SSSSSSSS SSSSSSSS

Table of contents

```
                        .TITLE  TTYCHARO - Terminal driver character output routine
;MIR0001                .IDENT  'V04-001'
-1
                        ;
                        ;*****************************************************************************
                        ;*                                                                           *
                        ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
                        ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
                        ;*  ALL RIGHTS RESERVED.                                                     *
                        ;*                                                                           *
                        ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
                        ;*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE           *
                        ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER      *
                        ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
                        ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY       *
                        ;*  TRANSFERRED.                                                             *
                        ;*                                                                           *
                        ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE     *
                        ;*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT         *
                        ;*  CORPORATION.                                                             *
                        ;*                                                                           *
                        ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS      *
                        ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
                        ;*                                                                           *
                        ;*****************************************************************************
                        ;
                        ;++
                        ; Facility:
                        ;
                        ;       VAX/VMS TERMINAL DRIVER
                        ;
                        ; ABSTRACT:
                        ;
                        ;       THIS MODULE CONTAINS ROUTINES USED FOR THE OUTPUT OF CHARACTERS.
                        ;
                        ;       ***** This module never destroys the contents of R0 *****
                        ;
                        ; AUTHOR:
                        ;
                        ;       R.HEINEN 11-AUG-1976
                        ;
                        ; Revision history:
                        ;
;MIR0001                ;       V04-001 MIR0001         Michael I. Rosenblum            05-Aug-1984
;MIR00C1                ;               Fix problem with Xon getting sent when a read with 0 timeout
;MIR0001                ;               gets completed with no charcters
;MIR0001                ;
                        ;       V03-046 MIR01424        Michael I. Rosenblum            16-Jun-1984
                        ;               Fix bug with read with 0 timeout that creates the typeahead
                        ;               buffer, reported in qar 1424.
                        ;
                        ;       V03-045 MIR0450         Michael I. Rosenblum            27-JUN-1984
                        ;               Make the port bit pc_nocrlf set the state skiplf
                        ;               Fix bug in escape introducer code that caused it not to
                        ;               recognize csi.
                        ;
                        ;       V03-044 MIR0400         Michael I. Rosenblum            10-Apr-1984
```

```
0000     54 ;
0000     55 ;          Filter null characters from escape sequences.
0000     56 ;
0000     57 ;  V03-043 MIR0390        Michael I. Rosenblum          04-Apr-1984
0000     58 ;          Make the exit string echo thru the tables vector.
0000     59 ;
0000     60 ;  V03-042 MIR0370        Michael I. Rosenblum          20-Mar-1984
0000     61 ;          Fix problems with tabs to the left of the curent position,
0000     62 ;          Lines that wrap in either the prompt or initial string.
0000     63 ;          Add port control word modifier to stop free linefeeds
0000     64 ;          After typed <CR>.
0000     65 ;
0000     66 ;  V03-042 MIR0310        Michael I. Rosenblum          07-Feb-1984
0000     67 ;          Fix bugs.
0000     68 ;          Noformat reads shouldn't wrap.
0000     69 ;          Ignore SKIPCRLF and EDITREAD if we are wrapping.
0000     70 ;          Don't change esc to $ if EDITREAD is set.
0000     71 ;          Move to beginning of line should rewrite the whole line
0000     72 ;            if the origional cursor position was non-zero.
0000     73 ;          Read nofltr should obey uppercase rules.
0000     74 ;          Passall should not imply termnoecho.
0000     75 ;          Fix delete in left justified read verify noecho fields
0000     76 ;            to not echo.
0000     77 ;
0000     78 ;  V03-041 MIR0300        Michael I. Rosenblum          30-Jan-1984
0000     79 ;          Make the uparrow key be command recall.
0000     80 ;
0000     81 ;  V03-040 MIR0100        Michael I. Rosenblum          24-Oct-1983
0000     82 ;          Fix bug that would cause characters to be echoed in no-echo
0000     83 ;          mode.
0000     84 ;
0000     85 ;  V03-039 MIR0084        Michael I. Rosenblum          25-Aug-1983
0000     86 ;          Fix problem with terminating a right justified numeric
0000     87 ;          auto-tab field in read verify.
0000     88 ;
0000     89 ;  V03-038 MIR0082        Michael I. Rosenblum          02-Aug-1983
0000     90 ;          Fix bug in recall logic that did not force a return before
0000     91 ;          the recall was done.  Fix bug in delete after wrap logic.
0000     92 ;
0000     93 ;  V03-037 MIR0081        Michael I. Rosenblum          01-Aug-1983
0000     94 ;          Remove $brddef call (obsoleted).
0000     95 ;
0000     96 ;  V03-036 MIR0080        Michael I. Rosenblum          28-Jul-1983
0000     97 ;          Reposition code in the module.
0000     98 ;
0000     99 ;  V03-035 MIR0070        Michael I. Rosenblum          13-Jul-1983
0000    100 ;          Fix bug that would cause an escape sequence at the beginning
0000    101 ;          of the line to return invalid data if deleted.
0000    102 ;          Make check in MOVEREADATA for buffer full condition at
0000    103 ;          the top of the loop to make sure that the users buffer
0000    104 ;          can not be overfilled if the initial string fills the
0000    105 ;          data buffer.
0000    106 ;
0000    107 ;  V03-034 MIR0053        Michael I. Rosenblum          27-Jun-1983
0000    108 ;          Fix bug in ESCAPE_TAB where it would pass the address
0000    109 ;          into the ESCINIT routine rather than the character
0000    110 ;
0000    111 ;  V03-033 MIR0051        Michael I. Rosenblum          23-Jun-1983
```

```
0000   111 ;                Fix bug caused when escape sequences are UNUSED.
0000   112 ;                Cause simceol to exit to movecursor on hard-copy terminals.
0000   113 ;                Make routines that checked CSI and ESC call a common routine.
0000   114 ;                Fix bug that would cause writes on non-hardcopy terminals
0000   115 ;                To use the single character code path following the wrap point.
0000   116 ;
0000   117 ;        V03-032 MIR0053         Michael I. Rosenblum            10-May-1983
0000   118 ;                Fix bug in clear to end of line emulation.
0000   119 ;
0000   120 ;        V03-031 MIR0052         Michael I. Rosenblum            27-May-1983
0000   121 ;                Add delete character to read verify, make read verify
0000   122 ;                accept fill characters in the initial string and as
0000   123 ;                data and display the clear character.
0000   124 ;                Add software emulation of clear to end of line for non-
0000   125 ;                ansi terminals.
0000   126 ;
0000   127 ;        V03-030 MIR0050         Michael I. Rosenblum            11-May-1983
0000   128 ;                Remove code that specail cased broadcasts.
0000   129 ;
0000   130 ;        V03-029 MIR0049         Michael I. Rosenblum            06-May-1983
0000   131 ;                Add code to handle wrapping correctly.
0000   132 ;
0000   133 ;        V03-028 MIR0041         Michael I. Rosenblum            29-Apr-1983
0000   134 ;                Fix bug in the new passall code path
0000   135 ;
0000   136 ;        V03-027 MIR0030         Michael I. Rosenblum            30-Mar-1983
0000   137 ;                Integrate TTRDVFY with the terminal driver, code included
0000   138 ;                in this module
0000   139 ;
0000   140 ;        V03-026 MIR0029         Michael I. Rosenblum            22-Mar-1983
0000   141 ;                Add code for insert/overstrike mode.
0000   142 ;
0000   143 ;        V03-025 RKS0025         RICK SPITZ                      19-MAR-1983
0000   144 ;                Change references of UCB$L_TT_DEVDP1 TO UCB$L_DEVDEPND2
0000   145 ;
0000   146 ;        V03-024 MIR6026         Michael I. Rosenblum            11-Mar-1983
0000   147 ;                Change recall to not terminate a small read when a recall
0000   148 ;                is done from a long recall buffer.
0000   149 ;
0000   150 ;        V03-023 MIR2026         Michael I. Rosenblum            07-Mar-1983
0000   151 ;                Fix bug in EDITREAD prompt echoing when there is no prompt
0000   152 ;                to echo.
0000   153 ;
0000   154 ;        V03-022 MIR1026         Michael I. Rosenblum            1-Mar-1983
0000   155 ;                Add $TRMDEF to the list of required symbol definitions
0000   156 ;
0000   157 ;        V03-021 MIR0026         Michael I. Rosenblum            10-Feb-1983
0000   158 ;                Add code to impliment input line editing.
0000   159 ;
0000   160 ;        V03-020 MIR0025         Michael I. Rosenblum            18-Feb-1983
0000   161 ;                Fix bug in carriage control logic that caused a carriage return
0000   162 ;                in a prompted read to get a free linefeed.
0000   163 ;
0000   164 ;        V03-019 MIR0024         Michael I. Rosenblum            28-Jan-1983
0000   165 ;                Change code to reflect redefinition of the read packet.
0000   166 ;
0000   167 ;        V03-018 MIR0017         Michael I. Rosenblum            05-Jan-1983
```

```
0000   168 ;               Add powerfail bit to the unit state vector table to remove
0000   169 ;               powerfail check for each character.  Restructure powerfail
0000   170 ;               code to fall into a common code path in TTYSUB.  Change
0000   171 ;               return status of TTY$GETNEXTCHAR to include a byte value
0000   172 ;               in the UCB, this will move the information from the condition
0000   173 ;               code bits.
0000   174 ;
0000   175 ;       V03-017 MIR0016         Michael I. Rosenblum            29-Dec-1982
0000   176 ;               Replace time calculation code with TIMSET macro call
0000   177 ;               This change will allow us to change the time calculation
0000   178 ;               algorhythm globaly if necessary, and will also allow
0000   179 ;               us to globaly turn on and off the time calculations.
0000   180 ;
0000   181 ;       V03-016 MIR0015         Michael I. Rosenblum            22-Dec-1982
0000   182 ;               Change all calls to the terminal port drivers to refer
0000   183 ;               to the class jacket routines. Remove DMA code from the
0000   184 ;               class driver.
0000   185 ;
0000   186 ;       V03-015 MIR0013         Michael I. Rosenblum            16-Dec-1982
0000   187 ;               Fix up refferences to new ucb structure
0000   188 ;
0000   189 ;       V03-014 MIR0011         Michael I. Rosenblum            18-Nov-1982
0000   190 ;               Change MULTIECHO to take a count in UCB$W_TT_MULTILEN and
0000   191 ;               an address in UCB$L_TT_MULTI, illiminate the code for
0000   192 ;               zero terminated multiecho strings.
0000   193 ;               Change STRTMULTI to take a counted string and convert
0000   194 ;               it into a length and address.
0000   195 ;               Add STRTMULTI_1 to take the place of STRTMULTI.
0000   196 ;               Add code to make Control-R and Control-U look prety on
0000   197 ;               ANSI crt terminals.
0000   198 ;               Change CONTROLR to EDITREAD also cnange the internals of
0000   199 ;               CONTROLR to use MULTIECHO.
0000   200 ;               Change the meaning of the CTRLR bit to indicate that the
0000   201 ;               prompt and data string from a read buffer are being clocked
0000   202 ;               out by multiecho, this is the only use for CTRLR.
0000   203 ;               Add SKIPCRLF to indicate that a linefeed following a <CR>
0000   204 ;               in the beginning of the prompt string is to be skiped.
0000   205 ;               Remove HOLDSCREEN code.
0000   206 ;
0000   207 ;       V03-013 MIR0012         Michael I. Rosenblum    19-Nov-1982
0000   208 ;               Fix bug that caused exception when a read was posted
0000   209 ;               that caused a create of the typeahead buffer.
0000   210 ;
0000   211 ;       V03-012 MIR0010         Michael I. Rosenblum    09-Nov-1982
0000   212 ;               Move the address of the terminator mask, and the length
0000   213 ;               of the prompt string from the IRP into the terminal read
0000   214 ;               packet.  Also move the count of the characters in the
0000   215 ;               buffer from the UCB into the terminal typeahead buffer packet.
0000   216 ;               Make backspace look like delete (without beeing delete)
0000   217 ;
0000   218 ;       V03-011 RKS0011         RICK SPITZ                      19-OCT-1982
0000   219 ;               INSURE THAT R5 IS PRESERVED WHEN DMA FORK TAKEN
0000   220 ;
0000   221 ;       V03-009 RKS0010         RICK SPITZ                      23-SEP-1982
0000   222 ;
0000   223 ;               CHANGE LOGIC IN FORMAT TO INSURE THAT HARDCOPY TERMINALS
0000   224 ;               SET TO NO WRAP DO NOT OUTPUT DATA BEYOND LINE WIDTH
```

```
0000    225 ;
0000    226 ;               CORRECT PROBLEM WITH XON LOGIC TO INSURE THAT OVERFLOW
0000    227 ;               ERRORS ARE ALWAYS REPORTED
0000    228 ;
0000    229 ;       V03-008 MIR0003         Michael I. Rosenblum    12-Aug-1982
0000    230 ;               Fix bug where multi escape sequences in a TTRDVFY prompt
0000    231 ;               Would not work.
0000    232 ;
0000    233 ;       V03-007 KDM0002         Kathleen D. Morse       28-Jun-1982
0000    234 ;               Added $SSDEF.
0000    235 ;--
```

```
                        .SBTTL  Declarations

;
; EXTERNAL SYMBOLS
;

        $CRBDEF                         ; DEFINE CRB
        $IODEF                          ; DEFINE I/O FUNCTION CODES
        $IPLDEF                         ; DEFINE IPL'S
        $IRPDEF                         ; DEFINE IRP
        $PRDEF                          ; DEFINE PROCESSOR REGISTERS
        $PRIDEF                         ; DEFINE PRIORITY CLASSES
        $RSNDEF                         ; DEFINE RESOURCE NUMBERS
        $SSDEF                          ; DEFINE SYSTEM STATUS CODES
        $UBADEF                         ; DEFINE UBA OFFSETS
        $UCBDEF                         ; DEFINE UCB
        $TRMDEF                         ; TERMINAL ITEMLIST BIT DEFINITIONS
        $TTYDEF                         ; DEFINE TERMINAL DRIVER SYMBOLS
        $TTDEF                          ; DEFINE TERMINAL CHARACTERISTICS
        $TT2DEF                         ; DEFINE EXTENDED TERMINAL CHARS
        $VECDEF                         ; DEFINE CRB VECTOR
        $TTYMACS                        ; DEFINE TERMINAL MACROS
        $TTYDEFS                        ; DEFINE TERMINAL DEFINITIONS

        .PSECT  $$$115_DRIVER,LONG      ; DEFINE NON-PAGED PSECT
```

```
                     0000   264          .SBTTL  TTY$GETNEXTCHAR - GET NEXT CHARACTER(S)
                     0000   265 ;++
                     0000   266 ; TTY$GETNEXTCHAR - GET NEXT CHARACTER(S)
                     0000   267 ;
                     0000   268 ; FUNCTIONAL DESCRIPTION:
                     0000   269 ;
                     0000   270 ; THIS ROUTINE RETURNS WITH THE NEXT CHARACTER(S) TO BE OUTPUT ON THE UNIT.
                     0000   271 ;
                     0000   272 ; INPUTS:
                     0000   273 ;
                     0000   274 ;       R5 = UCB ADDRESS
                     0000   275 ;
                     0000   276 ; OUTPUTS:
                     0000   277 ;
                     0000   278 ;       R0 = ALWAYS PRESERVED!!
                     0000   279 ;
                     0000   280 ;       R3 =    0       AND CC = ZERO - NO CHARACTER TO OUTPUT
                     0000   281 ;               CHAR    AND CC = PLUS - SINGLE CHARACTER TO OUTPUT
                     0000   282 ;               ADDRESS AND CC = NEG  - BURST (R2 = LENGTH)
                     0000   283 ;                                       (ADDRESS AND LENGTH ALSO IN UCB)
                     0000   284 ;
                     0000   285 ;       R5 = UCB ADDRESS
                     0000   286 ;       UCB$B_TT_OUTYPE = -1 BIRST
                     0000   287 ;                                  ADDRESS IN R3 AND UCB$L_TT_OUTADR
                     0000   288 ;                                  LENGTH IN R2  AND UCB$W_TT_OUTLEN
                     0000   289 ;                               0 NO CHARACTER TO OUTPUT
                     0000   290 ;                               1 SINGLE CHARACTER TO OUTPUT IN R3
                     0000   291 ;--
                     0000   292
                     0000   293 TTY$GETNEXTCHAR::
 52   0088 C5   9E   0000   294          MOVAB   UCB$Q_TT_STATE(R5),R2   ; ADDRESS STATE OF UNIT
                     0005   295 ;
                     0005   296 ; GET CURRENT STATE OF THE OUTPUT
                     0005   297 ;
                     0005   298 GETNEXTCHAR:                             ; INTERNAL ENTRY
 00000080 8F   D1   0005   299          CMPL    #TTY$M_ST_WRITE,(R2)    ; TRY WRITING GENERAL CASE FIRST
          62
          3F   13   0008   300          BEQL    WRITING                 ; IF EQL THEN GO
 00001000 8F   D1   000A   301          CMPL    #TTY$M_ST_READ,(R2)     ; TRY READING GENERAL CASE NEXT
          62
          03   12   0011   302          BNEQ    10$                     ; NO, GO TO 10$
        0513   31   0013   303          BRW     MOVEREADATA             ; IF EQL THEN GO
                     0016   304 10$:
 62   0D   00   EA   0016   305          FFS     #0,#TTY$V_ST_READ+1,(R2),R4; GET THE NEXT MOST IMPORTANT OUTPUT STAT
          54
                     001A   306          CASE    R4,TYPE=B,<-            ; ENTER THAT STATE'S ROUTINE
                     001A   307          POWERREST,-                     ; go to the power fail code
                     001A   308          INTEXIT,-                       ; CONTROL S
                     001A   309          FILLING,-                       ; FILL IN PROGESS
                     001A   310          CURSOROVRFLOW,-                 ; CURSOR OVERFLOW
                     001A   311          SENDLINEFEED,-                  ; LINE FEED TO BE INSERTED
                     001A   312          BACKSPACING,-                   ; BACKING UP IN PROGRESS
                     001A   313          MULTIECHOING,-                  ; MULTI ECHO IS PROGRESS
                     001A   314          WRITING,-                       ; WRITE IN PROGRESS
                     001A   315          EOLSEEN,-                       ; EOL SEEN
                     001A   316          EDITREAD,-                      ; READ EDITING OUTPUT STATE
                     001A   317          RDVERIFY,-                      ; GO TO READ VERIFICATION
```

```
                     001F   318          RECALLING,-                     ; RECALLING THE LAST COMMAND
                     001F   319          MOVEREADATA>                    ; READ IN PROGRESS
                     003D   320 ;
                     003D   321 ; EXIT INTERRUPT
                     003D   322 ;
                     003D   323 INTEXIT:                                  ; EXIT INTERRUPT
010B C5    94        003D   324          CLRB    UCB$B_TT_OUTYPE(R5)     ; SET NO RETURN CHARACTER
           05        0041   325          RSB                             ; RETURN
                     0042   326 ;
                     0042   327 ; POWERFAIL DETECTED
                     0042   328 ;
                     0042   329 POWERREST:
  FFBB'    30        0042   330          BSBW    RESTART                 ; CALL THE POWERFAIL RESTART CODE THEN
  FFBD     31        0045   331          BRW     GETNEXTCHAR             ; CONTINUE AGAIN
                     0048   332
                     0048   333
                     0048   334 INTEXIT1:                                ; EXIT AND RESTORE UCB ADDRESS
    55   8ED0        0048   335          POPL    R5
    F0     11        004B   336          BRB     INTEXIT
```

```
                        004D    338          .SBTTL  WRITING - WRITE state action routine
                        004D    339
                        004D    340 ;++
                        004D    341 ; WRITING - CONTINUE WRI.. I/O OUTPUT
                        004D    342 ;
                        004D    343 ; FUNCTIONAL DESCRIPTION:
                        004D    344 ;
                        004D    345 ; THIS ROUTINE IS ENTERED WHEN ALL EXTRA OUTPUT IS COMPLETE
                        004D    346 ; AND A WRITE OPERATION IS IN PROGRESS.
                        004D    347 ;
                        004D    348 ; THE NEXT AVAILABLE CHARACTER IS EXTRACTED FROM THE USER BUFFER
                        004D    349 ; AND GIVEN TO THE ECHO FORMAT ROUTINES TO OUTPUT CORRECTLY.
                        004D    350 ;
                        004D    351 ; IF CONTROL O HAS STOPPED THE OUTPUT THE OPERATION IS COMPLETED.
                        004D    352 ;
                        004D    353 ; INPUTS:
                        004D    354 ;
                        004D    355 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                        004D    356 ;       R5 = UCB ADDRESS
                        004D    357 ;
                        004D    358 ; Implicit inputs:
                        004D    359 ;
                        004D    360 ;       UCB$L_TT_WRTBUF - address of write buffer
                        004D    361 ;
                        004D    362 ; OUTPUTS:
                        004D    363 ;
                        004D    364 ;       R2 = ADDRESS OF THE TERMINAL STATE VECTOR
                        004D    365 ;       R3 = CHARACTER TO OUTPUT IF ANY
                        004D    366 ;       R5 = UCB ADDRESS
                        004D    367 ;--
                        004D    368
                        004D    369          .ENABLE LSB
                        004D    370 WRITING:
54   00D4 C5    D0      004D    371          MOVL    UCB$L_TT_WRTBUF(R5),R4  ; Get address of buffer block.
                        0052    372          IF_STATE CTRLO,60$              ; COMPLETE I/O OF CONTROL O
  53   1C A4    D0      0056    373          MOVL    TTY$L_WB_NEXT(R4),R3    ; GET ADDRESS OF NEXT CHARACTER
  20 A4   53    D1      005A    374          CMPL    R3,TTY$L_WB_END(R4)     ; DONE?
          09    1E      005E    375          BGEQU   50$                     ; IF GEQU THEN YES
                        0060    376          IF_STATE <PASALL,WRTALL>,BURST
        0074    31      0066    377          BRW     FORMAT                  ; FORMAT FOR OUTPUT
                        0069    378
                        0069    379 ;
                        0069    380 ; WRITE I/O DONE
                        0069    381 ;
                        0069    382
                        0069    383 50$:                                    ; Normal return.
                        0069    384          IF_NOT_STATE    <PASALL,WRTALL>,55$
    00FC C5     B4      006F    385          CLRW    UCB$W_TT_CURSOR(R5)     ; AT CONCLUSION OF PASSALL/WRITALL
    00FF C5     94      0073    386          CLRB    UCB$B_TT_LASTC(R5)      ; STATE IS UNKNOWN
                        0077    387 55$:
          01    B0      0077    388          MOVW    #SS$_NORMAL,-           ; Load success status code into
  28 A4                 0079    389                  TTY$W_WB_STATUS(R4)     ; buffer header.
          06    11      007B    390          BRB     70$                     ; CONTINUE
                        007D    391
                        007D    392 ;
                        007D    393 ; CONTROL O TYPED OR CONTROL Y OR C
                        007D    394 ;
```

```
                                L 11
                     007D   395
                     007D   396 60$:                                    ; Control-O.
         0609 8F     B0  007D   397          MOVW    #SS$_CONTROLO,-     ; Load control-O status code
           28 A4         0081   398                  TTY$W_WB_STATUS(R4) ; into buffer header.
                         0083   399
                         0083   400 70$:                                ; Complete transfer.
             1C A4       C3  0083   401          SUBL3   TTY$L_WB_NEXT(R4),-  ; Calculate number of characters
     53      20 A4           0086   402                  TTY$L_WB_END(R4),R3  ; not output to terminal.
     2A A4       53      A2  0089   403          SUBW    R3,TTY$W_WB_BCNT(R4) ; Adjust transfer count.
               FF70'     30  008D   404          BSBW    TTY$WRITEDONE       ; Do I/O done.
               FF6D      31  0090   405          BRW     TTY$GETNEXTCHAR     ; Get the next character.
                         0093   406
                         0093   407
                         0093   408 WRITE_END:
                         0093   409          IF_STATE_CTRLO,60$             ; CONTROL O
                   EA    11  0097   410          BRB     70$                ; COMPLETE WITH CURRENT STATUS
                         0099   411
                         0099   412          .DISABLE        LSB
                         0099   413
```

```
                                0099   415          .SBTTL BURST/STRING - SET UP OUTPUT FOR BURST MODE
                                0099   416 ;++
                                0099   417 ; BURST/STRING -- SET UP OUTPUT FOR BURST MODE
                                0099   418 ;
                                0099   419 ; FUNCTIONAL DESCRIPTION:
                                0099   420 ;
                                0099   421 ; THIS ROUTINE RETURNS THE ADDRESS AND LENGTH OF THE CURRENT
                                0099   422 ; OUTPUT TO THE CALLER. DUETIM IS COMPUTED BASED ON THE LENGTH OF THE
                                0099   423 ; OUTPUT. THE CURRENT WRITE POINTERS ARE UPDATED TO REFLECT THE
                                0099   424 ; NUMBER OF CHARACTERS INCLUDED IN THE BURST.
                                0099   425 ;
                                0099   426 ; INPUTS:
                                0099   427 ;       R3 = ADDRESS OF NEXT CHARACTER
                                0C99   428 ;       R4 = TWP ADDRESS
                                0099   429 ;       R5 = UCB ADDRESS
                                0099   430 ;
                                0099   431 ; OUTPUTS:
                                0099   432 ;
                                0099   433 ;       R3 = ADDRESS OF NEXT CHARACTER
                                0099   434 ;       R5 = UCB ADDRESS
                                0099   435 ;       UCB$L_TT_OUTADR = ADDRESS OF START OF BURST
                                0099   436 ;       UCB$L_TT_OUTLEN = LENGHT OF BURST
                                0099   437 ;--
                                0099   438 BURST:
      20 A4    53    C3       0099   439          SUBL3   R3,TTY$L_WB_END(R4),R2          ; COMPUTE LENGTH OF RECORD
               52
                                009D
                                009E   440
                                009E   441 STR_EXIT:
         52    52    3C       009E   442          MOVZWL  R2,R2                           ; CONVERT TO WORD VALUE
      1C A4    52    C0       00A1   443          ADDL    R2,TTY$L_WB_NEXT(R4)            ; UPDATE NEXT CHARACTER ADDRESS
    0120 C5    52    B0       00A5   444          MOVW    R2,UCB$W_TT_OUTLEN(R5)          ; SET OUTPUT SIZE
    011C C5    53    D0       00AA   445          MOVL    R3,UCB$L_TT_OUTADR(R5)          ; SET OUTPUT ADDRESS
                                00AF   446
                                00AF   447 STR_TIMESET:
                                00AF   448          TIMSET  R2,R1,LOCKOUTPUT                ; COMPUTE THE TIME PER CHARACTER
                                00D0   449                                                  ; AND SETUP INTERUPTS
                010B C5    01   8E   00D0   450          MNEGB   #1,UCB$B_TT_OUTYPE(R5)      ; SET NEGATIVE CC TO SIGNAL
                                00D5   451                                                  ; STRING OUTPUT
                                05   00D5   452          RSB
;MIR0001                        00D6    .1 STOP_NOW:
;MIR0001         00000000'GF    16   00D6    .2          JSB     G^INI$BRK
;MIR0001                        05   00DC    .3          RSB
                                00DD   453
                                00DD   454
```

```
                                00DD    456          .SBTTL FORMAT - FORMAT STRING FOR OUTPUT
                                00DD    457 ;++
                                00DD    458 ; FORMAT -- FORMAT STRING FOR OUTPUT
                                00DD    459 ; FUNCTIONAL DESCRIPTION:
                                00DD    460 ;
                                00DD    461 ; THIS ROUTINE RETURNS THE ADDRESS AND LENGTH OF A PORTION OF THE
                                00DD    462 ; CURRENT FORMATTED OUTPUT STRING TO THE CALLER.
                                00DD    463 ; IF NO SIGNIFICANT POPTION OF THE STRING CAN BE FOUND, IT
                                00DD    464 ; DEFAULTS TO SINGLE CHARACTER FORMAT MODE.
                                00DD    465 ; DUETIM IS COMPUTED BASED ON THE LENGTH OF THE
                                00DD    466 ; OUTPUT. THE CURRENT WRITE POINTERS ARE UPDATED TO REFLECT THE
                                00DD    467 ; NUMBER OF CHARACTERS INCLUDED IN THE BURST.
                                00DD    468 ;
                                00DD    469 ;
                                00DD    470 ; INPUTS:
                                00DD    471 ;       R2 = ADDRESS OF UNIT STATE VECTOR
                                00DD    472 ;       R3 = ADDRESS OF NEXT CHARACTER
                                00DD    473 ;       R4 = TWP ADDRESS
                                00DD    474 ;       R5 = UCB ADDRESS
                                00DD    475 ;
                                00DD    476 ; OUTPUTS:
                                00DD    477 ;
                                00DD    478 ;       R2 = ADDRESS OF UNIT STATE VECTOR
                                00DD    479 ;       R3 = ADDRESS OF NEXT CHARACTER
                                00DD    480 ;       R5 = UCB ADDRESS
                                00DD    481 ;       UCB$L_TT_OUTADR = ADDRESS OF START OF BURST
                                00DD    482 ;       UCB$L_TT_OUTLEN = LENGHT OF BURST
                                00DD    483 ;
                                00DD    484 ;                 OR
                                00DD    485 ;       R3 = NEXT CHARACTER TO OUTPUT
                                00DD    486 ;--
                                00DD    487 FORMAT:
                          50    DD    00DD    488          PUSHL   R0                              ; CSR MUST ALWAYS BE PRESERVED
                                00DF    489          IF_STATE <ESC_O>,FORMAT_X               ; IF ESCAPE IN PROGRESS, SPECIAL
                       20 A4    53    C3 00E3    490          SUBL3   R3,TT$L_WB_END(R4),R0           ; CALCULATE LENGTH OF STRING
                                50       00E7
                    05 44 A5    09    E0 00E8    491          BBS     #TT$V_WRAP,UCB$L_DEVDEPEND(R5),3$; IF WE ARE A NO-WRAP
                    14 44 A5    0C    E0 00ED    492          BBS     #TT$V_SCOPE,UCB$L_DEVDEPEND(R5),5$; SCOPE THEN OUTPUT THE MAXIMUM
                                00F2    493                                                  ; THAT WE CAN.
                       42 A5    01    A3 00F2    494 3$:      SUBW3   #1,UCB$W_DEVBUFSIZ(R5),-        ; COMPUTE EOL-1
                                51       00F6    495          R1
                     51    00FC C5    A2 00F7    496          SUBW2   UCB$W_TT_CURSOR(R5),R1          ; COMPUTE ROOM TILL EOL-1
                                45    1B 00FC    497          BLEQU   FORMAT_X                        ; EOL REACHED
                                00FE    498
                          51    51    32 00FE    499          CVTWL   R1,R1                           ;
                          50    51    D1 0101    500          CMPL    R1,R0                           ; OUTPUT SIZE IS LESS OF TWO
                                03    15 0104    501          BLEQ    10$
                          51    50    D0 0106    502 5$:      MOVL    R0,R1                           ; ACTUAL IS LESS
                                      0109    503
                                      0109    504 10$:
                       50    63    9A 0109    505          MOVZBL  (R3),R0                         ; TEST FIRST CHARACTER
                          F0 8F    93 010C    506          BITB    #<TTY$M_CH_CTRL!TTY$M_CH_SPEC!TTY$M_CH_CTRL2!TTY$M_CH_CTRL3>,-
                    0000'CF40          010F    507          W^TTY$A_TYPE[R0]
                                2E    12 0113    508          BNEQ    FORMAT_X
                                      0115    509
                                0E    BB 0115    510          PUSHR   #^M<R1,R2,R3>                   ; SAVE LENGTH, ADDRESS
                          63    51    2A 0117    511          SCANC   R1,(R3),W^TTY$A_TYPE,-          ; SKIP ALL NON SPECIALS
```

```
                0000'CF        011A
                  F0 8F        011D   512          #<TTY$M_CH_CTRL!TTY$M_CH_SPEC!TTY$M_CH_CTRL2!TTY$M_CH_CTRL3>
                               011F   513                                                ; TO ALLOW SINGLE BURST OUTPUT
                               011F   514 20$:
                      0E   BA  011F   515          POPR    #^M<R1,R2,R3>                 ; RESTORE LENGTH , ADDRESS
                51    50   C2  0121   516          SUBL    R0,R1                         ; COMPUTE NUMBER CHARS FOUND
                      1D   13  0124   517          BEQL    FORMAT_X                      ; YES
              00FC C5   51   A0  0126   518          ADDW    R1,UCB$W_TT_CURSOR(R5)        ; UPDATE FINAL CURSOR
                               012B   519                                                ; THIS ASSUMES ALL CHARS
                               012B   520                                                ; < SPACE ARE SPECIALS
                               012B   521          CLR_STATE <NL,SKIPLF,WRAP>            ;
                52    51   D0  0130   522          MOVL    R1,R2                         ; LENGTH OF STRING TO OUTPUT
                      50 8ED0  0133   523          POPL    R0                            ; RESTORE
          51    52    01   C3  0136   524          SUBL3   #1,R2,R1                      ; CALC N-1
      00FF C5     6341   90  013A   525          MOVB    (R3)[R1],UCB$B_TT_LASTC(R5)   ; LASTC IS FINAL CHARACTER OUTPUT
                  FF5B   31  0140   526          BRW     STR_EXIT                      ; TAKE COMMON STRING EXIT
                               0143   527
                               0143   528 FORMAT_X:
                      50 8ED0  0143   529          POPL    R0                            ; RESTORE
                53    63   9A  0146   530          MOVZBL  (R3),R3                       ; GET CHARACTER
                  1C A4   D6  0149   531          INCL    TTY$L_WB_NEXT(R4)             ; BUMP NEXT POINTER
                     003F   31  014C   532          BRW     FORMAT_CHAR                   ; HANDLE AS SINGLE
```

```
                     014F    534          .SBTTL  OUTPUTANDWAIT - OUTPUT CHARACTER AND WAIT FOR INTERRUPT
                     014F    535 ;++
                     014F    536 ; OUTPUTANDWAIT - OUTPUT A CHARACTER AND WAIT INTERRUPT
                     014F    537 ;
                     014F    538 ; FUNCTIONAL DESCRIPTION:
                     014F    539 ;
                     014F    540 ; THIS ROUTINE IS USED BY THE OUTPUT INTERRUPT ROUTINES TO RETURN
                     014F    541 ; TO THE DEVICE DEPENDENT CODE TO OUTPUT A CHARACTER. THEIR RETURN
                     014F    542 ; CAUSES THE UNIT TO ENTER A WAIT FOR INTERRUPT STATE.
                     014F    543 ;
                     014F    544 ; INPUTS:
                     014F    545 ;
                     014F    546 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                     014F    547 ;       R3 = CHARACTER TO OUTPUT
                     014F    548 ;       R5 = UCB ADDRESS
                     014F    549 ;
                     014F    550 ; OUTPUTS:
                     014F    551 ;
                     014F    552 ;       THE UNIT IS PLACED IN A WAIT FOR INTERRUPT STATE
                     014F    553 ;       AND THE CONDITION CODES ARE SET TO PLUS INDICATING
                     014F    554 ;       SINGLE CHARACTER IN R3.
                     014F    555 ;
                     014F    556 ;       R5 = UCB ADDRESS
                     014F    557 ;--
                     014F    558 OUTPUTANDWAIT:                          ; OUTPUT CHARACTER AND WAIT ENTRY
00FF C5   53   90    014F    559          MOVB    R3,UCB$B_TT_LASTC(R5)  ; SAVE LAST CHARACTER OUTPUT
                     0154    560 OUTPUTANDWAIT1:
                     0154    561          TIMSET  1,,LOCKOUTPUT          ; ENABLE THE TIMER WITH A MINIMUM
                     016E    562                                         ; OF 1 SECOND AND INTERLOCK THE
                     016E    563                                         ; OUTPUT STREAM
010B C5   01   90    016E    564          MOVB    #1,UCB$B_TT_OUTYPE(R5) ; SET THE KIND OF OUTPUT
          05         0173    565          RSB                            ; AND RETURN WITH CHARACTER
                     0174    566
```

```
                     017F   568          .SBTTL  FORMAT_CHAR - FORMAT CHARACTER FOR OUTPUT
                     0174   569
                     0174   570          .ENABL  LSB
                     0174   571 ;++
                     0174   572 ; FORMAT_CHAR - FIND PROPER OUTPUT FOR SPECIFIED CHARACTER
                     0174   573 ;
                     0174   574 ; FUNCTIONAL DESCRIPTION:
                     0174   575 ;
                     0174   576 ; THIS ROUTINE TRANSLATES THE SPECIFIED CHARACTER FOR OUTPUT
                     0174   577 ; ON THE TARGET UNIT. THE OUTPUT OF THE SEQUENCE IS EITHER
                     0174   578 ; THE ORIGIONAL CHARACTER OR A STARTUP OF THE PROPER MULTIECHO
                     0174   579 ; STRING. CURSOR ADJUSTMENT IS ALSO DONE HERE FOR PRINTING
                     0174   580 ; CHARACTERS AND FORM CHARACTERS. IT IS POSSIBLE FOR THE RESULT TO BE NO OUTPUT.
                     0174   581 ;
                     0174   582 ; SEE EACH SPECIAL CHARACTER ROUTINE FOR MORE DETAILS ON ECHOING.
                     0174   583 ;
                     0174   584 ; INPUTS:
                     0174   585 ;
                     0174   586 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                     0174   587 ;       R3 = CHARACTER TO TRANSLATE
                     0174   588 ;       R5 = UCB ADDRESS
                     0174   589 ;
                     0174   590 ; OUTPUTS:
                     0174   591 ;
                     0174   592 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                     0174   593 ;       R3 = CHARACTER TO OUTPUT NEXT IF ANY
                     0174   594 ;       R5 = UCB ADDRESS
                     0174   595 ;--
                     0174   596 5$:      IF_NOT_STATE <ESC_O>,OUTPUTANDWAIT; BR IF NOT OUTPUT ESC
           034F   30 0178   597          BSBW    ESCSYNTAX_O             ; CHECK CHAR AGAINST ESC SYNTAX
             D2   14 017B   598          BGTR    OUTPUTANDWAIT           ; SYNTAX OK, OUTPUT CHARACTER
                     017D   599          IF_NOT_STATE <MULTI>,8$         ; IF MULTIECHOING AND EDITING
                     0181   600          IF_STATE <EDITING>,10$          ; DON'T ZERO CURSOR
        00FC C5   B4 0185   601 8$:      CLRW    UCB$W_TT_CURSOR(R5)     ; ZERO CURSOR (COLUMN) POSITION
     04 A2    0E   E4 0189   602 10$:     BBSC    #TTY$V_ST_ESC_O,4(R2),- ; SEQUENCE ENDED CORRECTLY
             7A       018D   603                  GOOUT                   ; CLEAR STATE AND BRANCH ALWAYS
                     018E   604 FORMAT_CHAR:
                     018E   605          IF_STATE <PASALL,ESC_O,WRTALL>,5$       ; BR IF FORMAT NOT NEEDED
                     0196   606          CASE    W^TTY$A_TYPE[R3],LIMIT=#1@TTY$V_CH_SPEC,TYPE=B,<-
                     0196   607                          BSPACE,-
                     0196   608                          TAB,-
                     0196   609                          LINEFEED,-
                     0196   610                          VTAB,-
                     0196   611                          FORM,-
                     0196   612                          CARRIAGE,-
                     0196   613                          CTRLZ>
                     01AB   614
           A1 AF   9F 01AB   615          PUSHAB  OUTPUTANDWAIT           ; SET RETURN PC FOR ADJUST_CURSOR
                     01AE   616                                          ; AND FLOW INTO ADJUST_CURSOR
                     01AE   617
                     01AE   618          .DSABL  LSB
                     01AE   619
```

```
                     01AE   621          .SBTTL  ADJUST_CURSOR - Increment cursor and set wrap if needed
                     01AE   622
                     01AE   623 ADJUST_CURSOR:                              ; ADJUST CURSOR
                     01AE   624
       F0 8F    93   01AE   625          BITB    #<TTY$M_CH_CTRL!TTY$M_CH_SPEC!TTY$M_CH_CTRL2!TTY$M_CH_CTRL3>,-
 0000'CF43          01B1   626          W^TTY$A_TYPE[R3]                       ; TEST FOR SPECIAL
          28    12   01B5   627          BNEQ    30$                            ; NON SPACING CHARACTER
    00FC C5    B6   01B7   628          INCW    UCB$W_TT_CURSOR(R5)            ; ADJUST CURSOR
    00FC C5    B1   01BB   629          CMPW    UCB$W_TT_CURSOR(R5),UCB$W_DEVBUFSIZ(R5) ; OVERRUN?
       42 A5          01BF
          06    1E   01C1   630          BGEQU   5$                             ; YES
                     01C3   631
                     01C3   632 25$:     CLR_STATE -                            ; Set not at newline,
                     01C3   633                  <NL,SKIPLF,WRAP>              ; clear SKIPLF and WRAP.
                     01C8   634 26$:     RSB
                     01C9   635
                     01C9   636 5$:
 0C 44 A5    09    E0   01C9   637          BBS     #TT$V_WRAP,UCB$L_DEVDEPEND(R5),10$; BR IF WRAP ENABLED
          F3    13   01CE   638          BEQL    25$                            ; IF EQL THEN AT END OF LINE AND OK
 EE 44 A5    0C    E0   01D0   639          BBS     #TT$V_SCOPE, -                 ; IF SCREEN TERMINAL,CONTINUE
                     01D5   640                  UCB$L_DEVDEPEND(R5),25$ ;
             8E    D5   01D5   641          TSTL    (SP)+                          ; DO NOT RETURN TO BSB TO DROP
           00C6    31   01D7   642          BRW     DROP                           ; IF HARDCOPY,DROP CHARACTER
                     01DA   643 10$:     SET_STATE       CURSOR                 ; SET CURSOR OVERFLOWED
             E4    11   01DD   644          BRB     25$
                     01DF   645
             04    91   01DF   646 30$:     CMPB    #TTY$K_ET_ESCAPE,TTY$A_CCLIST[R3]; IS THIS AN ESCAPE OR CSI
 00000000'EF43      01E1
             DF    12   01E7   647          BNEQ    26$                            ; NO THEN DON'T COUNT IT
             8E    D5   01E9   648          TSTL    (SP)+                          ; DO NOT RETURN TO BSB TO DROP
           00DE    31   01EB   649          BRW     ESCAPE                         ; *** FALL INTO THE ESCAPE CODE
```

```
                        .SBTTL  FORMAT_LOCAL - FORMAT CHARACTER FOR OUTPUT

;++
; FORMAT_LOCAL - FIND PROPER OUTPUT FOR SPECIFIED CHARACTER
;
; FUNCTIONAL DESCRIPTION:
;
; THIS ROUTINE TRANSLATES THE SPECIFIED CHARACTER FOR OUTPUT
; ON THE TARGET UNIT FOR A NOECHO TERMINAL. THE OUTPUT OF THE
; SEQUENCE MAY BE THE STARTUP OF THE PROPER MULTIECHO
; STRING. OTHERWISE, CURSOR ADJUSTMENT IS ALSO DONE HERE FOR PRINTING
; CHARACTERS AND FORM CHARACTERS.
; IT IS POSSIBLE FOR THE RESULT TO BE NO OUTPUT.
;
; SEE EACH SPECIAL CHARACTER ROUTINE FOR MORE DETAILS ON ECHOING.
;
; INPUTS:
;
;       R2 = ADDRESS OF THE UNIT STATE VECTOR
;       R3 = CHARACTER TO TRANSLATE
;       R5 = UCB ADDRESS
;
; OUTPUTS:
;
;       R2 = ADDRESS OF THE UNIT STATE VECTOR
;       R3 = CHARACTER TO OUTPUT NEXT IF ANY
;       R5 = UCB ADDRESS
;--

FORMAT_LOCAL:
        CASE    W^TTY$A_TYPE[R3],LIMIT=#1@TTY$V_CH_SPEC,TYPE=B,<-
                        BSPACE,-
                        TAB_LOCAL,-
                        LINEFEED,-
                        VTAB,-
                        FORM,-
                        CARRIAGE,-
                        CTRLZ>
;
; SPECIAL FORMATTING WAS NOT NEEDED - SIMPLY INCREMENT CURSOR
; FOR THE LOCALLY-ECHOED CHARACTER AND CONTINUE.
;
        BSBB    ADJUST_CURSOR           ; ADJUST CURSOR
        BRW     GETNEXTCHAR             ; GET ANOTHER CHARACTER
GOOUT:  BRW     OUTPUTANDWAIT
```

```
                     020B   697          .SBTTL  CHARACTER OUTPUT FORMAT ACTION ROUTINES
                     020B   698          .SBTTL  BACKSPACE - output a backspace
                     020B   699
                     020B   700 ;++
                     020B   701 ; BACKSPACE - OUTPUT A BACKSPACE
                     020B   702 ;
                     020B   703 ; FUNCTIONAL DESCRIPTION:
                     020B   704 ;
                     020B   705 ; THIS ROUTINE OUTPUTS A BACKSPACE ON  A TERMINAL.
                     020B   706 ;
                     020B   707 ; INPUTS:
                     020B   708 ;
                     020B   709 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                     020B   710 ;       R3 = TTY$C_BS
                     020B   711 ;       R5 = UCB ADDRESS
                     020B   712 ;
                     020B   713 ; OUTPUTS:
                     020B   714 ;
                     020B   715 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                     020B   716 ;       R3 = TTY$C_BS
                     020B   717 ;       R4 = ADDRESS OF THE BS/SPACE/BS STRING IF APPROP.
                     020B   718 ;       R5 = UCB ADDRESS
                     020B   719 ;--
                     020B   720 BSPACE:                                      ; BACKSPACE
00FC C5    B7        020B   721          DECW    UCB$W_TT_CURSOR(R5)        ; ADJUST HORIZON
   FF3D    31        020F   722          BRW     OUTPUTANDWAIT              ; OUTPUT THE CHARACTER
```

```
                     0212   724          .SBTTL  CARRIAGE - format a carriage return
                     0212   725
                     0212   726 ;++
                     0212   727 ; CARRIAGE - FORMAT FOR CARRIAGE RETURN
                     0212   728 ;
                     0212   729 ; FUNCTIONAL DESCRIPTION:
                     0212   730 ;
                     0212   731 ; THIS ROUTINE SETS UP THE PROPER FILL FOR A CARRIAGE RETURN ON
                     0212   732 ; THE TARGET UNIT.
                     0212   733 ;
                     0212   734 ; INPUTS:
                     0212   735 ;
                     0212   736 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                     0212   737 ;       R3 = TTY$C_CR
                     0212   738 ;       R5 = UCB ADDRESS
                     0212   739 ;
                     0212   740 ; OUTPUTS:
                     0212   741 ;
                     0212   742 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                     0212   743 ;       R3 = TTY$C_CR
                     0212   744 ;       R5 = UCB ADDRESS
                     0212   745 ;--
                     0212   746 CARRIAGE:                                    ;
                     0212   747          CLR_STATE <NL>                     ; SET NOT AT NEW LINE.
                     0216   748 CROUTPUT:
                     0216   749          IF_STATE .                         ; If in a write state, branch
                     0216   750                   WRITE,115$                ; forward.
                     021A   751          IF_NOT_STATE -                     ; If not in a read state, also
                     021A   752                   READ,115$                 ; branch forward.
                     021E   753          IF_STATE <MULTI,EDITREAD>,130$     ; not a character echo or wrap
  51    78 A5    D0  0225   754          MOVL    UCB$L_SVAPTE(R5),R1        ; GET THE PACKET ADDRESS
  54    30 A1    3C  0229   755          MOVZWL  TTY$W_RB_LINOFF(R1),R4     ; GET THE OFFSET TO THIS CHARACTER
     2C B144    9E  022D   756          MOVAB   @TTY$L_RB_LIN(R1)[R4],TTY$L_RB_LIN(R1); AND MAKE THIS THE NEW
       2C A1         0231
       30 A1    B4  0233   757          CLRW    TTY$W_RB_LINOFF(R1)        ; LINE BEGININNG
       32 A1    B4  0236   758          CLRW    TTY$W_RB_LINREST(R1)       ; JUST MAKE SURE WE ARE AT THE END
0122 C5    07    E1  0239   759          BBC     #TTY$V_PC_NOCRLF,UCB$W_TT_PRTCTL(R5),110$;SO JUST ECHO THE CHARACTER
                   3C  023E
                     023F   760          SET_STATE SKIPLF                   ; NO FREE LINEFEED
           41    11  0243   761          BRB     115$                       ; ELSE DON'T DO THE LINEFEED
                     0245   762
                     0245   763 130$:    IF_NOT_STATE CTRLR,110$            ;
                     0249   764 ;
                     0249   765 ; THE FOLLOWING CODE WILL INSERT A LINEFEED AFTER ANY CARRIAGE RETURN
                     0249   766 ; THAT IS THE LAST CHARACTER IN THE READ PROMPT STRING OR IN THE READ BUFFER
                     0249   767 ; ITSELF.  ANY <CR> CHARACTERS IN THE PROMPT STRING PRIOR TO THE LAST CHARACTER
                     0249   768 ; ARE NOT TOUCHED.
                     0249   769 ;
  51    78 A5    D0  0249   770          MOVL    UCB$L_SVAPTE(R5),R1        ; GET THE READ PACKET ADDRESS
  54    34 A1    3C  024D   771          MOVZWL  TTY$W_RB_PRMLEN(R1),R4     ; GET THE PROMPT LENGTH
54    4A A144    9E  0251   772          MOVAB   TTY$A_RB_PRM(R1)[R4],R4    ; AND THIS IS THE ADDRESS
54    00D8 C5    D1  0256   773          CMPL    UCB$L_TT_MULTI(R5),R4      ; IS THIS CR PAST END OF PROMPT?
           22    1E  025B   774          BGEQU   112$                       ; YES THEN FORCE LINEFEED
                     025D   775 ; In the case where we are sending a clear to end of line sequence (signaled
                     025D   776 ; by SKIPCRLF being set) then
                     025D   777 ; we don't want the linefeed to show up if there is one after a leading
                     025D   778 ; <CR>.  The case where there is a leading linefeed is handled by
```

```
                     025D   779 ; setting SKIPLF in the EDITREAD routine.
                     025D   780 ;
                     025D   781         IF_NOT_STATE SKIPCRLF,115$       ; IF NO LINEFEEDS WANTED THEN
                     0261   782         CLR_STATE <SKIPCRLF>             ; CLEAR NO LINEFEED
 0000004B 8F    C1   0265   783         ADDL3   #TTY$L_RB_DATA+1,UCB$L_SVAPTE(R5),R4; GET THE ADDRESS OF THE
   54    78 A5       026B
54   00D8 C5    D1   026E   784         CMPL    UCB$L_TT_MULTI(R5),R4    ; FIRST CHARACTER AND SEE IF THIS IS IT
           11    12   0273   785         BNEQU   115$                     ; NO THEN HANDLE NORMALY
                      0275   786         SET_STATE <SKIPLF>               ; OTHERWISE SKIP THE LINEFEED
           0B    11   0279   787         BRB     115$                     ; AND HANDLE NORMALY.
                      027B   788
                      027B   789 110$:
                      027B   790         IF_STATE <SKIPCRLF>,115$
                      027F   791 112$:   SET_STATE <SENDLF,SKIPLF>        ; SET STATE TO FORCE LF AND SKIP NEXT REAL O
                      0286   792 115$:
      00FC C5    B4   0286   793         CLRW    UCB$W_TT_CURSOR(R5)      ; RESET HORIZON
0D    00FF C5    91   028A   794         CMPB    UCB$B_TT_LASTC(R5), -    ; OPTIMIZATION: WAS LAST CHAR
                      028F   795                 #TTY$C_CR                ; A CARRIAGE RETURN?
           0F    13   028F   796         BEQL    DROP                     ; YES, DON'T OUTPUT ANOTHER
      00F6 C5    90   0291   797         MOVB    UCB$B_TT_CRFILL(R5),UCB$B_TT_FILL(R5); SET UP FILL
      0102 C5         0295
           03    13   0298   798         BEQL    120$                     ; IF EQL THEN OUTPUT
                      029A   799         SET_STATE FILL                   ; OTHERWISE OUTPUT
              FEAF   31   029D   800 120$:  BRW     OUTPUTANDWAIT
              FD62   31   02A0   801 DROP:  BRW     GETNEXTCHAR              ; CONTINUE IN NEXT LOGICAL STATE
```

```
                        02A3    803          .SBTTL  CTRLZ - output control-Z
                        02A3    804
                        02A3    805 ;++
                        02A3    806 ; CTRLZ - OUTPUT A CONTROL Z BASED ON THE OPERATION
                        02A3    807 ;
                        02A3    808 ; FUNCTIONAL DESCRIPTION:
                        02A3    809 ;
                        02A3    810 ; IF THE CURRENT OPERATION IS A READ AND THE ^Z IS A TERMINATOR THE ECHO ^Z.
                        02A3    811 ;
                        02A3    812 ; INPUTS:
                        02A3    813 ;
                        02A3    814 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                        02A3    815 ;       R3 = CONTROL Z
                        02A3    816 ;       R5 = UCB ADDRESS
                        02A3    817 ;
                        02A3    818 ; OUTPUTS:
                        02A3    819 ;
                        02A3    820 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                        02A3    821 ;       R3 = CONTROL Z
                        02A3    822 ;       R4 = ADDRESS OF THE MULTIECHO STRING FOR CONTROL Z IF APPROP.
                        02A3    823 ;       R5 = UCB ADDRESS
                        02A3    824 ;--
                        02A3    825 CTRLZ:                                      ; CONTROL Z
                        02A3    826         IF_STATE WRITE,300$                 ; IF THERE IS A WRITE ACTIVE THEN
                        02A7    827                                             ; ECHO NORMALY
                        02A7    828         IF_NOT_STATE READ,300$              ; OUTPUT 26(8) IF NOT READ
                        02AB    829         IF_NOT_STATE EOL,300$               ; ^Z IF TERMINATOR
00000000'EF    D0       02AF    830         MOVL    TTY$A_EXITECHO,R4           ; ADDRESS MULTIECHO STRING
          54
                        02B5
08 48 A5       1D       E1      02B6    831         BBC     #TT2$V_DECCRT,UCB$L_DEVDEPND2(R5),299$
                        02BB    832         SET_STATE NINTMULTI                 ; DON'T ALLOW THIS TO BE INTERUPTED
   54    04 A4          DE      02BF    833         MOVAL   4(R4),R4            ; OUTPUT THE DECCRT CONTROL-Z
      54    64          D0      02C3    834 299$:   MOVL    (R4),R4             ; GET THE ACTUAL STRING ADDRESS
            0DA2          31    02C6    835         BRW     STRTMULTI           ; START OUTPUT
            FE83          31    02C9    836 300$:   BRW     OUTPUTANDWAIT
```

```
                     02CC    838          .SBTTL  ESCAPE - format an escape character
                     02CC    839
                     02CC    840 ;++
                     02CC    841 ; ESCAPE - FORMAT A ESCAPE BASED ON OPERATION AND TARGET UNIT
                     02CC    842 ;
                     02CC    843 ; FUNCTIONAL DESCRIPTION:
                     02CC    844 ;
                     02CC    845 ; THIS ROUTINE FORMATS ESCAPES.
                     02CC    846 ;
                     02CC    847 ; READ OPERATION:
                     02CC    848 ;
                     02CC    849 ;       A "$" IS ECHOED IF THE ESCAPE IS A TERMINATOR.
                     02CC    850 ;
                     02CC    851 ; WRITE OPERATION:
                     02CC    852 ;
                     02CC    853 ;       THE ESCAPE IS OUTPUT. ON TERMINALS WITH THE CHARACTERISTIC TT$M_ESCAPE,
                     02CC    854 ;       THE REMAINDER OF THE SEQUENCE IS CHECKED FOR SYNTACTIC CORRECTNESS.
                     02CC    855 ;
                     02CC    856 ; INPUTS:
                     02CC    857 ;
                     02CC    858 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                     02CC    859 ;       R3 = ESCAPE
                     02CC    860 ;       R5 = UCB ADDRESS
                     02CC    861 ;
                     02CC    862 ; OUTPUTS:
                     02CC    863 ;
                     02CC    864 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                     02CC    865 ;       R3 = ESCAPE OR "$"
                     02CC    866 ;       R5 = UCB ADDRESS
                     02CC    867 ;--
                     02CC    868 ESCAPE:                                     ; OUTPUT ESCAPE STRING
                     02CC    869         IF_NOT_STATE EOL,250$               ; ALSO IF NOT TERMINATOR
                     02D0    870         IF_STATE <MULTI,WRITE,EDITREAD>,250$; MULTIECHO THEN ASSUME SEQUENCE
          53   24    9A    02D7    871         MOVZBL  #TTY$C_DOLLAR,R3        ; OTHERWISE ECHO DOLLAR SIGN
             FED1    30    02DA    872         BSBW    ADJUST_CURSOR           ; ADJUST CURSOR
             FE6F    31    02DD    873         BRW     OUTPUTANDWAIT           ; AND OUTPUT
                     02E0    874 ;
                     02E0    875 ; OUTPUT ESCAPE ON WRITE OR CONTROL R
                     02E0    876 ;
                     02E0    877 250$:
16 48 A5     18     E0    02E0    878         BBS     #TT2$V_ANSICRT,UCB$L_DEVDEPND2(R5),255$ ; BR IF ANSI TERMINAL
                     02E5    879         IF_STATE <ESCAPE>,255$              ; IS THIS ESCAPE MODE
41 A5    40 8F    91    02E9    880         CMPB    #TT$_VT5X,UCB$B_DEVTYPE(R5) ;VT5X OR VT100 TERMINAL?
               02    18    02EE    881         BLEQU   252$                    ; If LEQU, then maybe.
               23    11    02F0    882         BRB     260$                    ; Otherwise, no.
                     02F2    883
                     02F2    884 252$:
41 A5    80 8F    91    02F2    885         CMPB    #TT$_VT100+32,UCB$B_DEVTYPE(R5);
               02    1A    02F7    886         BGTRU   255$                    ; If GTRU, then yes.
               1A    11    02F9    887         BRB     260$                    ; Otherwise, no.
                     02FB    888
                     02FB    889 255$:
                     02FB    890         SET_STATE <ESC_0>                   ; OUTPUTTING AN ESCAPE SEQUENCE
               51    DD    0300    891         PUSHL   R1                      ; SAVE R1
             019E    30    0302    892         BSBW    ESCINIT                 ; INIT THE ESCAPE SEQUENCE RULES
0104 C5        51    90    0305    893         MOVB    R1,UCB$B_TT_ESC_0(R5)   ; ...
               51  8ED0    030A    894         POPL    R1                      ; RESTORE R1
```

```
                     030D   895
                     030D   896          IF_NOT_STATE <MULTI>,260$       ; MULTIECHOING AND EDITING
                     0311   897          IF_STATE <EDITING>,270$         ; DON'T ZERO CURSOR
                     0315   898 260$:
00FC C5   B4         0315   899          CLRW    UCB$W_TT_CURSOR(R5)     ; ZERO CURSOR (COLUMN) POSITION
    FE33   31        0319   900 270$:    BRW     OUTPUTANDWAIT           ; FALL INTO OUTPUT AND WAIT
```

```
                        031C   902          .SBTTL  LINEFEED - format a line feed
                        031C   903
                        031C   904 ;++
                        031C   905 ; LINEFEED - FORMAT LINE FEED FOR TARGET UNIT
                        031C   906 ;
                        031C   907 ; FUNCTIONAL DESCRIPTION:
                        031C   908 ;
                        031C   909 ; THIS ROUTINE SETS UP THE PROPER FILL FOR A LINE FEED ON THE TARGET
                        031C   910 ; UNIT AND ADJUSTS THE CURSOR AND VERTICAL LINE COUNT.
                        031C   911 ;
                        031C   912 ; INPUTS:
                        031C   913 ;
                        031C   914 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                        031C   915 ;       R3 = TTY$C_LF
                        031C   916 ;       R5 = UCB ADDRESS
                        031C   917 ;
                        031C   918 ; OUTPUTS:
                        031C   919 ;
                        031C   920 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                        031C   921 ;       R3 = TTY$C_LF
                        031C   922 ;       R5 = UCB ADDRESS
                        031C   923 ;--
                        031C   924          .ENABLE LSB
                        031C   925 LINEFEED:
            3B 04 A2 0D E4 031C   926          BBSC    #TTY$V_ST_SKIPLF,4(R2),227$; SKIP AND SET OFF SKIP CONDITION
                        0321   927 ;
                        0321   928 ; NOTE THAT BECAUSE OF ACBB'S SIGNED BRANCH IT IS NOT APPROPRIATE HERE.
                        0321   929 ;
                        0321   930 LFOUTPUT:
              00FC C5 B5 0321   931          TSTW    UCB$W_TT_CURSOR(R5)     ; LINE FEED AT CURSOR 0 IS NEWLINE
                 04 12 0325   932          BNEQ    210$                    ; IF NEQ THEN NO NL POSSIBLE
                        0327   933          SET_STATE NL                    ; SET NEW LINE
              00FE C5 96 032B   934 210$:    INCB    UCB$B_TT_LINE(R5)       ; ADJUST VERTICAL COUNT
              00FE C5 91 032F   935          CMPB    UCB$B_TT_LINE(R5),UCB$L_DEVDEPEND+3(R5);
                47 A5        0333
                 04 1F 0335   936          BLSSU   215$                    ; IF LSSU THEN NO PAGE CROSS
              00FE C5 94 0337   937          CLRB    UCB$B_TT_LINE(R5)       ; RESET VERTICAL POSITION
                        033B   938 215$:    IF_NOT_STATE WRITE,220$         ; BR IF NOT WRITE
          54  00D4 C5 D0 033F   939          MOVL    UCB$L_TT_WRTBUF(R5),R4  ; Address current write buffer.
            54  24 A4 D0 0344   940          MOVL    TTY$L_WB_IRP(R4),R4     ; Get associated IRP address.
                 03 13 0348   941          BEQL    220$                    ; Branch if no IRP.
                38 A4 B6 034A   942          INCW    IRP$L_MEDIA(R4)         ; COUNT LINE
              00F7 C5 90 034D   943 220$:    MOVB    UCB$B_TT_LFFILL(R5),UCB$B_TT_FILL(R5); INSERT TO CAUSE FILL
              0102 C5        0351
                 03 13 0354   944          BEQL    225$                    ; OUTPUT IF NO FILL NEEDED
                        0356   945          SET_STATE FILL                  ; OUTPUT FILL CHARACTERS
                 FDF3 31 0359   946 225$:    BRW     OUTPUTANDWAIT
                 FF41 31 035C   947 227$:    BRW     DROP
                        035F   948          .DISABLE LSB
```

```
                        035F    950          .SBTTL  TAB - Output a tab.
                        035F    951
                        035F    952 ;++
                        035F    953 ; TAB - OUTPUT A TAB ON THE TARGET TERMINAL BASED ON CURSOR POSITION
                        035F    954 ;
                        035F    955 ; FUNCTIONAL DESCRIPTION:
                        035F    956 ;
                        035F    957 ; THIS ROUTINE IS ENTERED TO OUTPUT A TAB ON THE TARGET UNIT.
                        035F    958 ; IF THE TERMINAL HAS MECHANICAL TAB THEN THE TAB GOES DIRECT.
                        035F    959 ; OTHERWISE, THE CURSOR POSITION IS USED TO CALC. HOW MANY BLANKS
                        035F    960 ; TO OUTPUT TO MOVE THE CURSOR TO THE NEXT TAB STOP.
                        035F    961 ;
                        035F    962 ; INPUTS:
                        035F    963 ;
                        035F    964 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                        035F    965 ;       R3 = TTY$C_TAB
                        035F    966 ;       R5 = UCB ADDRESS
                        035F    967 ;
                        035F    968 ; OUTPUTS:
                        035F    969 ;
                        035F    970 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                        035F    971 ;       R3 = TTY$C_TAB
                        035F    972 ;       R4 = ADDRESS OF TAB MULTIECHO STRING IF APPROP.
                        035F    973 ;       R5 = UCB ADDRESS
                        035F    974 ;--
      03    00    EF    035F    975 TAB:    EXTZV   #0,#3,UCB$W_TT_CURSOR(R5),R4; GET HORIZON POINTER
54    00FC  C5          0362
                        0366    976         IF_STATE TABEXPAND,525$          ; EXPAND THE TABS
0C 44 A5    08    E0    036A    977         BBS     #TT$V_MECHTAB,UCB$L_DEVDEPEND(R5),530$; OUTPUT IF MECHAN.CAL HELP
51    08    54    A3    036F    978 525$:   SUBW3   R4,#8,R1                 ;SETUP THE LENGTH OF THE TAB
54    0001'CF   9E    0373    979         MOVAB   W^TTY$A_TAB+1,R4         ; ADDRESS STRING TO OUTPUT
            0CFE  31    0378    980         BRW     STRTMULTI_1              ; START MULTIPLE OUTPUT
                        037B    981 ;
                        037B    982 ; MECHANICAL TAB TERMINAL
                        037B    983 ;
                        037B    984 ; THIS ROUTINE ASSUMES THAT THE TABS ARE SET ON 8'S
                        037B    985 ;
41 A5       40 8F     91    037B    986 530$:   CMPB    #TT$_VT5X,UCB$B_DEVTYPE(R5); IN RANGE FOR SPECIAL CASE VT5X
                18    1A    0380    987         BGTRU   535$                     ; IF GTR THEN NO
41 A5       80 8F     91    0382    988         CMPB    #TT$_VT100+32,UCB$B_DEVTYPE(R5);
                11    1B    0387    989         BLEQU   535$                     ;
          00FC  C5    A3    0389    990         SUBW3   UCB$W_TT_CURSOR(R5),UCB$W_DEVBUFSIZ(R5),R3; DISTANCE FROM RIGHT
      53    42    A5          038D
                05    1B    0390    991         BLEQU   534$                     ; IF LEQU THEN SKIP TEST
            53    08    B1    0392    992         CMPW    #8,R3                    ; IN LAST TAB SPACE?
                  D8    1E    0395    993         BGEQU   525$                     ; IF TRUE THEN SKIP MECH TAB
            53    09    9A    0397    994 534$:   MOVZBL  #TTY$C_TAB,R3            ;
00FC  C5    54    AA    039A    995 535$:   BICW    R4,UCB$W_TT_CURSOR(R5)   ; SETUP CURSOR STOP ON 8
00FC  C5    08    A0    039F    996         ADDW    #8,UCB$W_TT_CURSOR(R5)   ;
                        03A4    997         CLR_STATE -                      ; No longer at newline; haven't
                        03A4    998                 <NL,WRAP>                ; just wrapped.
                FDA3    31    03A9    999         BRW     OUTPUTANDWAIT            ;
                        03AC    1000 ;
                        03AC    1001 ; FOR LOCAL TABS, TRACK POSITION, BUT DON'T ECHO
                        03AC    1002 ;
                        03AC    1003
                        03AC    1004 TAB_LOCAL:
```

```
                03    00    EF   03AC  1005          EXTZV   #0,#3,UCB$W_TT_CURSOR(R5),R4; GET HORIZON POINTER
54              00FC  C5         03AF
00FC C5         54    AA   03B3  1006          BICW    R4,UCB$W_TT_CURSOR(R5)  ; SETUP CURSOR STOP ON 8
00FC C5         08    A0   03B8  1007          ADDW    #8,UCB$W_TT_CURSOR(R5)  ;
                           03BD  1008          CLR_STATE -
                           03BD  1009                  <NL,WRAP>
                    FC40   31   03C2  1010          BRW     GETNEXTCHAR             ; GET NEXT OUTPUT CHARACTER
```

```
                     03C5  1012          .SBTTL  VTAB, FORM - output a vertical tab, or form feed
                     03C5  1013
                     03C5  1014 ;++
                     03C5  1015 ; VTAB - OUTPUT A VERTICAL TAB
                     03C5  1016 ; FORM - OUTPUT A FORM FEED
                     03C5  1017 ;
                     03C5  1018 ; FUNCTIONAL DESCRIPTION:
                     03C5  1019 ;
                     03C5  1020 ; THIS ROUTINE SETS UP THE PROPER OUTPUT STRING FOR A VTAB OF FORM FEED
                     03C5  1021 ; ON THE TARGET UNIT.  VTAB TRANSLATES TO 4 LINE FEEDS. AND FORM FEED
                     03C5  1022 ; TO MULTIPLE LINE FEEDS BASED ON THE PRESENCE OF MECHANICAL HELP
                     03C5  1023 ; AND THE CURRENT VERTICAL LINE POSITION. TO AVOID THE NECESSITY
                     03C5  1024 ; FOR A LARGE NUMBER OF LF DATA BYTES THE FORM FEED CAUSES A MULTI
                     03C5  1025 ; ECHO STRING OF 4 LINE FEEDS FOLLOWED BY ANOTHER FORM FEED TO BE
                     03C5  1026 ; OUTPUT UNTIL A PAGE CROSS TAKES PLACE.
                     03C5  1027 ;
                     03C5  1028 ; INPUTS:
                     03C5  1029 ;
                     03C5  1030 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                     03C5  1031 ;       R3 = C_VTAB OR C_FF
                     03C5  1032 ;       R5 = UCB ADDRESS
                     03C5  1033 ;
                     03C5  1034 ; OUTPUTS:
                     03C5  1035 ;
                     03C5  1036 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                     03C5  1037 ;       R5 = UCB ADDRESS
                     03C5  1038 ;--
          54 0000'CF 9E  03C5  1039 VTAB:   MOVAB   W^TTY$A_VTAB,R4          ; SET UP OUTPUT
               0C9E  31  03CA  1040         BRW     STRTMULTI               ; START MULTIPLE OUTPUT
                     03CD  1041 ;
                     03CD  1042 ; FORM FEED
                     03CD  1043 ;
07 44 A5   13  E1  03CD  1044 FORM:   BBC     #TT$V_MECHFORM, -       ; BRANCH IF TERMINAL DOFS NOT
                     03D2  1045                 UCB$L_DEVDEPEND(R5), -  ; SUPPORT MECHNICAL FORM FEEDS
                     03D2  1046                 505$                    ;
     00FE C5   94  03D2  1047         CLRB    UCB$B_TT_LINE(R5)       ; RESET LINE POSITION
        FD76   31  03D6  1048         BRW     OUTPUTANDWAIT           ;
          54   D4  03D9  1049 505$:   CLRL    R4                      ; SEND MULTIPLE LINE FEEDS
     00FE C5   83  03DB  1050         SUBB3   UCB$B_TT_LINE(R5),UCB$L_DEVDEPEND+3(R5),R4; GET NUMBER TO END OF PAG
   54   47 A5        03DF
     04   54   91  03E2  1051         CMPB    R4,#4                   ; OUTPUT THE LF'S IN GROUPS OF 8
          0C   1A  03E5  1052         BGTRU   520$                    ; IF GTRU THEN MORE THAN 4 TO OUTPUT
51   54   01   A1  03E7  1053 510$:   ADDW3   #1,R4,R1                ; GET THE NUMBER OF LINEFEEDS
54   0001'CF   9E  03EB  1054         MOVAB   W^TTY$A_FORM+1,R4       ; GET THE ADDRESS OF THE STRING
        0C86   31  03F0  1055         BRW     STRTMULTI_1             ; START OUTPUT
54   0000'CF   9E  03F3  1056 520$:   MOVAB   W^TTY$A_LONGFORM,R4      ; GET ADDRESS TO STRING
        0C70   31  03F8  1057         BRW     STRTMULTI               ;
```

```
                03FB  1059          .SBTTL  DISPATCHER SERVICE ROUTINES
```

```
                     03FB  1061          .SBTTL  BACKSPACING - OUTPUT N BACKSPACES
                     03FB  1062 ;++
                     03FB  1063 ; BACKSPACING:
                     03FB  1064 ;
                     03FB  1065 ; DESCRIPTION:
                     03FB  1066 ;       SEND A GIVEN NUMBER OF BACKSPACES TO THE TERMINAL
                     03FB  1067 ;
                     03FB  1068 ; INPUTS:
                     03FB  1069 ;       R2 = UNIT STATE VECTOR ADDRESS
                     03FB  1070 ;       R5 = UCB ADDRESS
                     03FB  1071 ;       UCB$W_TT_MULTILEN = NUMBER OF BACKSPACES TO OUTPUT
                     03FB  1072 ;--
                     03FB  1073 BACKSPACING:
0100 C5    B7        03FB  1074          DECW    UCB$W_TT_BSPLEN(R5)     ; SUBTRACT ONE AND OUTPUT THE BACKSPACE
     06    19        03FF  1075          BLSS    10$                     ; NO MORE THEN FINISH UP
53   08    9A        0401  1076          MOVZBL  #TTY$C_BS,R3            ; SETUP BACKSPACE
   FD87    31        0404  1077          BRW     FORMAT_CHAR             ; AND FORMAT THE CHARACTER
                     0407  1078 10$:     CLR_STATE BACKSPACE             ; CLEAR THE STATE WHEN DONE
   FBF8    31        040A  1079          BRW     GETNEXTCHAR             ; AND GET THE NEXT CHARACTER
                     040D  1080
```

```
                     040D  1082          .SBTTL  CURSOROVRFLOW - insert newline to handle end of line
                     040D  1083 ;++
                     040D  1084 ; CURSOROVRFLOW - INSERT A NEWLINE IN THE OUTPUT STREAM
                     040D  1085 ;
                     040D  1086 ; FUNCTIONAL DESCRIPTION:
                     040D  1087 ;
                     040D  1088 ; THIS ROUTINE IS ENTERED TO INJECT A CR/LF IN THE OUTPUT
                     040D  1089 ; STREAM REGARDLESS OF THE CURRENT STATE.
                     040D  1090 ;
                     040D  1091 ; INPUTS:
                     040D  1092 ;
                     040D  1093 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                     040D  1094 ;       R5 = UCB ADDRESS
                     040D  1095 ;
                     040D  1096 ; OUTPUTS:
                     040D  1097 ;
                     040D  1098 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                     040D  1099 ;       R3 = TTY$C_CR
                     040D  1100 ;       R4 = READ PACKET ADDRESS
                     040D  1101 ;       R5 = UCB ADDRESS
                     040D  1102 ;--
                     040D  1103 CURSOROVRFLOW:                                       ;
                     040D  1104          CLR_STATE CURSOR                            ; CLEAR THE CONDITION
                     0410  1105          SET_STATE WRAP                              ; Signal WRAPping.
                     0415  1106          IF_NOT_STATE READ,99$                       ; IF WE ARE NOT READING THEN HANDLE NORMALY
                     0419  1107          IF_STATE RDVERIFY,98$                       ; NO CURSOR ORERFLOW ON READ VERIFY'S
54   78 A5    D0    041D  1108          MOVL    UCB$L_SVAPTE(R5),R4                 ; OTHERWISE GET THE PACKED ADDRESS
                     0421  1109          IF_NOT_STATE CTRLR,97$                      ; IF WE ARE CONTROL-RING HANDLE NORMALY
   00D8 C5    D1    0425  1110          CMPL    UCB$L_TT_MULTI(R5),TTY$L_RB_LIN(R4); IF WE AREN'T ON THE LAST LINE
     2C A4
        05    1B    0429  1111          BLEQU   99$                                 ; THEN DON'T WORRIE ABOUT LINREST
     32 A4    B5    042B  1112 97$:     TSTW    TTY$W_RB_LINREST(R4)                ; ARE WE IN THE MIDDLE OF A LINE
        0D    12    042D  1113          BNEQ    98$                                 ; YES THEN LEAVE THE STATUS QUOE
53      0D    9A    0430  1114 99$:     MOVZBL  #TTY$C_CR,R3                        ; SET FIRST CHARACTER IS CARRIAGE RETURN
                     0432  1115          SET_STATE <SENDLF,SKIPLF>                   ; FORCE A FREE LINEFEED
      FDD7    31    0435  1116          BRW     CROUTPUT                            ; CONTINUE WITH CARRIAGE RETURN CODE
      FE5E    31    0438  1117 98$:     BRW     DROP
```

```
                        0442  1119          .SBTTL  EOLSEEN - handle end of line condition
                        0442  1120
                        0442  1121 ;++
                        0442  1122 ; EOLSEEN - END OF LINE SEEN
                        0442  1123 ;
                        0442  1124 ; FUNCTIONAL DESCRIPTION:
                        0442  1125 ;
                        0442  1126 ; THIS ROUTINE IS ENTERED AFTER AN END OF LINE CHARACTER HAS
                        0442  1127 ; TERMINATED A READ RECORD AND THE ECHO OF THAT CHARACTER IS COMPLETE.
                        0442  1128 ;
                        0442  1129 ; THE ACTION IS TO COMPLETE THE READ OPERATION AND IF NECESSARY,
                        0442  1130 ; SEND AN XOFF.
                        0442  1131 ;
                        0442  1132 ; INPUTS:
                        0442  1133 ;
                        0442  1134 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                        0442  1135 ;       R5 = UCB ADDRESS
                        0442  1136 ;
                        0442  1137 ; OUTPUTS:
                        0442  1138 ;
                        0442  1139 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                        0442  1140 ;       R5 = UCB ADDRESS
                        0442  1141 ;--
                        0442  1142 EOLSEEN:                                    ; END READ OPERATION
                        0442  1143 ;
                        0442  1144 ; IN ORDER FOR THE FINAL CHARACTERS OF A READ VERIFY READ TO ECHO
                        0442  1145 ; PROPERLY EDIT READ MUST BE ALLOWED TO COMPLETE, THIS CAN BE DONE
                        0442  1146 ; IN TWO WAYS, FIRST CHANGING THE STATE DISPATCHER TO ALLOW EDIT READ
                        0442  1147 ; TO HAVE HIGHER PRIORITY THEN EOL.  THIS FIXES EOL IN THE STATE TABLE
                        0442  1148 ; FOR NOW AND FOREVER.  BY PUTTING THIS CODE HERE IT ALLOWS EOL TO
                        0442  1149 ; MOVE TO ANY PLACE IN THE STATE DISPATCHER.
                        0442  1150 ;
                        0442  1151          IF_NOT_STATE RDVERIFY,5$          ; LET EDIT READ COMPLETE UNDER
                        0446  1152          IF_NOT_STATE EDITREAD,5$          ; READ VERIFY
              0909   31 044A  1153          BRW     EDITREAD                  ;...
                        044D  1154
7C A5   01    9B 044D  1155 5$:       MOVZBW  #SS$_NORMAL,UCB$W_BOFF(R5); SET STATUS
        FBAC'   30 0451  1156          BSBW    TTY$READONE               ; COMPLETE THE I/O
        FBA9    31 0454  1157          BRW     TTY$GETNEXTCHAR           ; CONTINUE
```

```
                     0457  1159          .SBTTL  FILLING - continue outputting fill characters
                     0457  1160
                     0457  1161 ;++
                     0457  1162 ; FILLING - CONTINUE FILL OUTPUT
                     0457  1163 ;
                     0457  1164 ; FUNCTIONAL DESCRIPTION:
                     0457  1165 ;
                     0457  1166 ; THIS ROUTINE IS ENTERED WHILE THE TRM$V_FILL STATE IS ON.
                     0457  1167 ; THE FILL COUNT IS DECREMENTED AND IF NON 0 A FILL IS OUTPUT
                     0457  1168 ; IF THE COUNT GOES TO 0 THEN THE FILL STATE IS CLEARED.
                     0457  1169 ;
                     0457  1170 ; INPUTS:
                     0457  1171 ;
                     0457  1172 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                     0457  1173 ;       R5 = UCB ADDRESS
                     0457  1174 ;
                     0457  1175 ; OUTPUTS:
                     0457  1176 ;
                     0457   177 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                     0457  1178 ;       R3 = 0
                     0457  1179 ;       R5 = UCB ADDRESS
                     0457  1180 ;--
                     0457  1181 FILLING:                                    ; OUTPUT CONTINUING FILLS
0102 C5    97        0457  1182          DECB    UCB$B_TT_FILL(R5)         ; ADJUST FILL COUNT
       03   12        045B  1183          BNEQ    25$                       ; IF NEQ THEN NOT DONE YET
                     045D  1184          CLR_STATE FILL                    ; SET CONDITION OFF
         53  D4        0460  1185 25$:     CLRL    R3                        ; SET FILL CHARACTER
     FCEF    31        0462  1186          BRW     OUTPUTANDWAIT1            ; OUTPUT THE CHARACTER
```

```
                     0465  1188          .SBTTL  MULTIECHOING - Continue outputting multiecho sequence
                     0465  1189
                     0465  1190 ;++
                     0465  1191 ; MULTIECHOING - CONTINUE MULTIECHO STRING OUTPUT
                     0465  1192 ;
                     0465  1193 ; FUNCTIONAL DESCRIPTION:
                     0465  1194 ;
                     0465  1195 ; THIS ROUTINE IS ENTERED WHEN TTY$V_ST_MULTI IS SET. THE NEXT
                     0465  1196 ; CHARACTER IN THE STRING ADDRESSED BY UCB$L_TT_MULTI IS OUTPUT.
                     0465  1197 ; IF THE NEXT CHARACTER IS ZERO THEN THE STRING OUTPUT IS COMPLETE
                     0465  1198 ; AND THE MULTIECHO CONDITION IS RESET FOLLOWED BY A RETURN TO
                     0465  1199 ; THE OUTPUT STATE ANALYSIS ROUTINE IN GETNEXTCHAR.
                     0465  1200 ; IF A LENGTH IS SPECIFIED IN UCB$W_TT_MULTILEN, THEN IT IS USED TO DETECT
                     0465  1201 ; THE END OF STRING.
                     0465  1202 ;
                     0465  1203 ; INPUTS:
                     0465  1204 ;
                     0465  1205 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                     0465  1206 ;       R5 = UCB ADDRESS
                     0465  1207 ;
                     0465  1208 ;       UCB$W_TT_MULTILEN = ALTERNATE LENGTH
                     0465  1209 ;
                     0465  1210 ; OUTPUTS:
                     0465  1211 ;
                     0465  1212 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                     0465  1213 ;       R3 = CHARACTER TO OUTPUT ( FALL THROUGH TO ECHOFORMAT )
                     0465  1214 ;       R5 = UCB ADDRESS
                     0465  1215 ;--
                     0465  1216 MULTIECHOING:                                ; CONTINUE STRING MULTI ECHO
00DC C5    B7        0465  1217          DECW    UCB$W_TT_MULTILEN(R5)      ; DECREMENT THE COUNT
      23   18        0469  1218          BGEQ    20$                        ; CONTINUE IF MORE LEFT
00E0 C5    D0        046B  1219          MOVL    UCB$L_TT_SMLT(R5),UCB$L_TT_MULTI(R5); RESTORE SAVED ADDRESS
00D8 C5              046F
00DE C5    B0        0472  1220          MOVW    UCB$W_TT_SMLTLEN(R5),UCB$W_TT_MULTILEN(R5);AND LENGTH
00DC C5              0476
      0B   14        0479  1221          BGTR    10$
                     047B  1222          CLR_STATE <MULTI,NINTMULTI>        ; NO MORE THEN RESET THE STATE
    FB7F   31        0483  1223          BRW     GETNEXTCHAR                ; AND GET THE NEXT CHARACTER
00DE C5    B4        0486  1224 10$:     CLRW    UCB$W_TT_SMLTLEN(R5)       ; CLEAR THE SAVED LENGTH
00DC C5    B7        048A  1225          DECW    UCB$W_TT_MULTILEN(R5)      ; SUBTRACT OUT THIS CHARACTER
53  00D8 D5    9A        048E  1226 20$:     MOVZBL  @UCB$L_TT_MULTI(R5),R3     ; GET NEXT MULTI ECHO CHARACTER
00D8 C5    D6        0493  1227          INCL    UCB$L_TT_MULTI(R5)         ; ADJUST POINTER
    FCF4   31        0497  1228          BRW     FORMAT_CHAR                ;
```

```
                                                J 13
                     049A  1230          .SBTTL  SENDLINEFEED - output a line feed
                     049A  1231
                     049A  1232 ;++
                     049A  1233 ; SENDLINEFEED - INSERT LINE FEED IN OUTPUT STREAM
                     049A  1234 ;
                     049A  1235 ; FUNCTIONAL DESCRIPTION:
                     049A  1236 ;
                     049A  1237 ; THIS ROUTINE IS USED TO INSERT A LINE FEED IN THE OUTPUT STREAM.
                     049A  1238 ;
                     049A  1239 ; INPUTS:
                     049A  1240 ;
                     049A  1241 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                     049A  1242 ;       R5 = UCB ADDRESS
                     049A  1243 ;
                     049A  1244 ; OUTPUTS:
                     049A  1245 ;
                     049A  1246 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                     049A  1247 ;       R3 = TTY$C_LF
                     049A  1248 ;       R5 = UCB ADDRESS
                     049A  1249 ;--
                     049A  1250 SENDLINEFEED:                        ; SEND OUT A LINE FEED
                     049A  1251          CLR_STATE SENDLF            ; CLEAR CONDITION
 53   0A    9A       049D  1252          MOVZBL  #TTY$C_LF,R3        ; SET TO OUTPUT LINE FEED
    FE7E    31       04A0  1253          BRW     LFOUTPUT            ; CONTINUE IN LINE FEED CODE
```

```
                                                K 13
                     04A3  1255          .SBTTL  ESCAPE SEQUENCE PARSING SERVICES
                     04A3  1256 ;++
                     04A3  1257 ;
                     04A3  1258 ; ESCINIT - INIT ESCAPE SEQUENCE RULES
                     04A3  1259 ;
                     04A3  1260 ; INPUTS:
                     04A3  1261 ;       R3 = CHARACTER THAT IS INTRODUCER
                     04A3  1262 ;
                     04A3  1263 ; OUTPUTS:
                     04A3  1264 ;       R1 = RULE TO START AT (BYTE)
                     04A3  1265 ;
                     04A3  1266 ESCINIT::
           51    D4  04A3  1267          CLRL    R1                              ; USE THE NORMAL RULES
        80 8F    53  91  04A5  1268          CMPB    R3,#^X80                        ; IF NOT AN 8 BIT CHARACTER
              08    1F  04A9  1269          BLSSU   10$                             ;
FFFFFF80'EF43    90  04AB  1270          MOVB    TTY$A_ESCINIT-^X80[R3],R1       ; ELSE MOVE THRU THE TABLE
           51        04B2
                 05  04B3  1271 10$:     RSB
```

```
                                        04B4  1273          .SBTTL  ESCSYNTAX -- CHECK ESCAPE SEQUENCE SYNTAX
                                        04B4  1274
                                        04B4  1275 ;++
                                        04B4  1276 ; FUNCTIONAL DESCRIPTION:
                                        04B4  1277 ;       This routine checks the syntax of escape sequences, one
                                        04B4  1278 ;       character at a time.
                                        04B4  1279 ;
                                        04B4  1280 ; INPUTS:
                                        04B4  1281 ;       R3 contains the next character to be checked for correct syntax
                                        04B4  1282 ;       UCB$B_TT_ESC(R5) contains a offset to the next syntax
                                        04B4  1283 ;                rule in the TTY$A_ESCAPE syntax table for input data.
                                        04B4  1284 ;       UCB$B_TT_ESC_O(R5) contains the offset for output data.
                                        04B4  1285 ;       Assume R4 is available
                                        04B4  1286 ;
                                        04B4  1287 ; OUTPUTS:
                                        04B4  1288 ;       Condition codes are set:
                                        04B4  1289 ;       CC = POSITIVE means the character is syntactically correct
                                        04B4  1290 ;       CC = EQUAL means syntactic correctness and sequence is complete
                                        04B4  1291 ;       CC = NEGATIVE means error in parsing sequence
                                        04B4  1292 ;
                                        04B4  1293 ;       If CC = POSITIVE the UCB$B_TT_ESC (ESC_O) offset is updated.
                                        04B4  1294 ;--
                                        04B4  1295 ;
                                        04B4  1296 ; INPUT ESCAPE SEQUENCES
                                        04B4  1297 ;
                                        04B4  1298
                                        04B4  1299 ESCSYNTAX:
                                  54    DD  04B4  1300          PUSHL   R4                      ; SAVE R4
                            54    0103  C5  04B6  1301          MOVZBL  UCB$B_TT_ESC(R5),R4     ; GET CURRENT STATE
                                  1A    10  04BB  1302          BSBB    E_SYNTAX
                            0103  C5    54  04BD  1303          MOVB    R4,UCB$B_TT_ESC(R5)     ; SAVE STATE
                                  54  8ED0  04C2  1304          POPL    R4                      ; AND RESTORE R4
                                0103  C5    95  04C5  1305          TSTB    UCB$B_TT_ESC(R5)        ; SET THE CONDITION CODES
                                      05    04C9  1306          RSB
                                            04CA  1307
                                            04CA  1308 ;
                                            04CA  1309 ; OUPUT ESCAPE SEQUENCES
                                            04CA  1310 ;
                                            04CA  1311
                                            04CA  1312 ESCSYNTAX_O:
                            54    0104  C5    9A  04CA  1313          MOVZBL  UCB$B_TT_ESC_O(R5),R4   ; GET CURRENT STATE
                                        06    10  04CF  1314          BSBB    E_SYNTAX
                            0104  C5    54    90  04D1  1315          MOVB    R4,UCB$B_TT_ESC_O(R5)   ; SAVE CURRENT STATE
                                        05    04D6  1316          RSB
                                            04D7  1317
                                            04D7  1318
                                            04D7  1319 ;
                                            04D7  1320 ; GENERAL SEQUENCE HANDLER
                                            04D7  1321 ;
                                            04D7  1322
                                            04D7  1323 E_SYNTAX::
                                            04D7  1324
                      0000'CF44     53    91  04D7  1325 10$:     CMPB    R3, W^TTY$A_ESCAPE[R4]  ; check range of character
                                    0F    1F  04DD  1326          BLSSU   20$                     ; branch if not this rule
                      0001'CF44     53    91  04DF  1327          CMPB    R3, W^TTY$A_ESCAPE+1[R4]; lower than high limit?
                                    07    1A  04E5  1328          BGTRU   20$                     ; look to next rule
                                            04E7  1329
```

```
                54   0002'CF44    9A  04E7  1330          MOVZBL  W^TTY$A_ESCAPE+2[R4],-  ; character is valid
                                       04ED  1331                  R4                    ; save offset to next rule
                                       04ED  1332
                                  05   04ED  1333          RSB                           ; offset=POS => next rule exists
                                       04EE  1334                                        ; offset=0 => end of sequence
                                       04EE  1335
                                       04EE  1336 ;
                                       04EE  1337 ; Continue looking for correct sequence rule
                                       04EE  1338 ;
                         0002'CF44     95  04EE  1339 20$:    TSTB    W^TTY$A_ESCAPE+2[R4]  ; any next rule?
                                 05    13  04F3  1340          BEQL    30$                   ; no; bad syntax
                           54    03    C0  04F5  1341          ADDL    #3, R4                ; offset to next rule
                                 DD    11  04F8  1342          BRB     10$                   ; continue syntax check
                                       04FA  1343 30$:
                           54    01    CE  04FA  1344          MNEGL   #1, R4                ; error in sequence
                                       05  04FD  1345          RSB
```

```
                                04FE  1347          .SBTTL MOVEREADATA - MOVE CHARACTER FROM TYPEAHEAD TO READ BUFFER
                                04FE  1348 ;++
                                04FE  1349 ; MOVEREADATA -- MOVE CHARACTER FROM TYPEAHEAD BUFFER TO READ BUFFER
                                04FE  1350 ;
                                04FE  1351 ; FUNCTIONAL DESCRIPTION:
                                04FE  1352 ;
                                04FE  1353 ; THIS ROUTINE MOVES A CHARACTER FROM THE TYPEAHEAD BUFFER AND STARTS
                                04FE  1354 ; THE ECHO.
                                04FE  1355 ;
                                04FE  1356 ; NON-IMMEDIATE ACTION CONTROL SEQUENCES ARE HANDLED HERE.
                                04FE  1357 ;
                                04FE  1358 ; BEFORE RETURNING A CHARACTER FOR ECHO IT IS CONVERTED TO ITS
                                04FE  1359 ; MULTIPLE ECHO STRING IF APPROPRIATE. IN THIS CASE THE CHARACTER
                                04FE  1360 ; RETURNED IS THE FIRST OF THE MULTIPLE ECHO CHARACTERS.
                                04FE  1361 ;
                                04FE  1362 ; INPUTS:
                                04FE  1363 ;       R5 = UCB ADDRESS
                                04FE  1364 ;
                                04FE  1365 ; OUTPUTS:
                                04FE  1366 ;
                                04FE  1367 ;       R3 = CHARACTER IF ANY (CC = EQL )
                                04FE  1368 ;       R5 = UCB ADDRESS
                                04FE  1369 ;--
                                04FE  1370 ;
                                04FE  1371 ; GOTO TABLE
                                04FE  1372 ;
        011F   31   04FE  1373 TAB_NOTA:       BRW     NOTA
51  58 A5   D0   0501  1374 TAB_EOLSEEN:    MOVL    UCB$L_IRP(R5),R1
    38 A1   B4   0505  1375                 CLRW    IRP$L_MEDIA(R1)
     FF37   31   0508  1376                 BRW     EOLSEEN
     0153   31   050B  1377 TAB_DISMISS:    BRW     DISMISS
     0101   31   050E  1378 TAB_BUFEMPTY:   BRW     BUFEMPTY
     0232   31   0511  1379 TAB_INDELST:    BRW     INDELST
     0C1F   31   0514  1380 TAB_PASSALL:    BRW     PASSALL
     FAEB   31   0517  1381 TAB_GETNEXT:    BRW     GETNEXTCHAR
     02DC   31   051A  1382 TAB_QUOTE:      BRW     QUOTE
     02EE   31   051D  1383 TAB_EDITINGCHAR:BRW     EDITINGCHAR
       53   95   0520  1384 TAB_ESCINPROG:  TSTB    R3                      ; NO NULLS IN ESCAPE SEQUENCFS
       E7   13   0522  1385                 BEQL    TAB_DISMISS
     013F   31   0524  1386                 BRW     ESCINPROG
     012A   31   0527  1387 TAB_CVTLOW:     BRW     CVTLOW
     0230   31   052A  1388 TAB_OPEN:       BRW     OPEN
                  052D  1389 ;
                  052D  1390 ; MAIN LINE MOVE OUT OF BUFFER CODE
                  052D  1391 MOVEREADATA:
                  052D  1392          .ENABLE LSB
  54  78 A5   D0   052D  1393          MOVL    UCB$L_SVAPTE(R5),R4     ; SETUP R4 TO CONTAIN THE ADDRESS OF THE
      14 A4   D4   0531  1394          CLRL    TTY$L_RB_ECHSTR(R4)     ; CLEAN OUT ANY UNUSED ECHO STRINGS
40 A4 3C A4   B1   0534  1395          CMPW    TTY$W_RB_TXTOFF(R4),TTY$W_RB_TXTSIZ(R4); TEST BUFFER FULL
        C6   1E   0539  1396          BGEQU   TAB_EOLSEEN             ; BUFFER FULL THEN END
                  053B  1397 ;
                  053B  1398 ; MOVE CHARACTER OUT OF TYPEAHEAD BUFFER
                  053B  1399 ;
  54 00E4 C5   D0   053B  1400          MOVL    UCB$L_TT_TYPAHD(R5),R4  ; ADDRESS TYPEAHEAD BUFFER
        BC   13   0540  1401          BEQL    TAB_NOTA                ; NO BUFFER THEN THERE IS A FORK
                  0542  1402                                          ; PROCESS WAITING SO LET IT GO.
      0C A4   B7   0542  1403          DECW    TTY$W_TA_INAHD(R4)      ; take the character out of typeahead
```

```
                C7   19  0545  1404          BLSS    TAB_BUFEMPTY            ; IF EQL THEN NO
          53    04 B4   9A  0547  1405          MOVZBL  @TTY$L_TA_GET(R4),R3    ; GET THE CHARACTER
                        054B  1406          IF_STATE DEL,TAB_INDELST        ; ARE WE IN DELETE STATE
                        054F  1407 ;
                        054F  1408 ; ACTUALLY REMOVE CHARACTER FROM BUFFER
                        054F  1409 ;
                        054F  1410 ;
                        054F  1411 ; CHECK FOR BUFFER WRAP AROUND
                        054F  1412 ;
    04 A4       10 A4   F2  054F  1413 40$:     AOBLSS  TTY$L_TA_END(R4),TTY$L_TA_GET(R4),60$; POINTER PAST END?
                        06  0554
          0118  C4   9E  0555  1414          MOVAB   TTY$L_TA_DATA(R4),TTY$L_TA_GET(R4); RESET POINTER
            04  A4       0559
                        055B  1415 ;
                        055B  1416 ; SKIP TESTS FOR SECOND CONTROL SEQUENCE SET IF PASSALL
                        055B  1417 ;
                        055B  1418
          54    78 A5   D0  055B  1419 60$:     MOVL    UCB$L_SVAPTE(R5),R4     ; SETUP R4 TO CONTAIN THE ADDRESS OF THE
                        055F  1420          IF_STATE <PASALL,NOFLTR>,TAB_PASSALL; IN PASSALL THEN PASS ALL
                        0566  1421          IF_STATE QUOTING,TAB_QUOTE      ; ARE WE QUOTING
00000000'EF43   9A  056A  1422 67$:     MOVZBL  TTY$A_CCLIST[R3],R1     ; GET THE TOKEN
              51        0571
              A9   12   0572  1423          BNEQ    TAB_EDITINGCHAR         ; SPECAIL ACTION THEN HANDLE SPECAILY
                        0574  1424 ;
                        0574  1425 ; INSERT THE CHARACTER
                        0574  1426 ;
                        0574  1427 151$:   IF_STATE <ESC>,TAB_ESCINPROG; IF IN AN ESCAPE SEQUENCE THEN HANDLE AS SUCH
                        0578  1428 INSERT_CHAR:
    51  58 A5   D0  0578  1429 152$:   MOVL    UCB$L_IRP(R5),R1        ; GET THE IRP ADDRESS
A6  20 A1    08   E0  057C  1430          BBS     #IO$V_CVTLOW,IRP$W_FUNC(R1),TAB_CVTLOW; BR IF CONVERT LOWER TO UPPER
3D  1C B4    53   E0  0581  1431 155$:   BBS     R3,@TTY$L_RB_TERM(R4),TERMFOUND; IS THIS CHARACTER A TERMINATOR
    51  30 A4   3C  0586  1432          MOVZWL  TTY$W_RB_LINOFF(R4),R1  ; GET THE OFFSET INTO THE BUFFER
        32 A4   B5  058A  1433          TSTW    TTY$W_RB_LINREST(R4)     ; DO WE HAVE TO OPEN THE BUFFER
           9B   12  058D  1434          BNEQ    TAB_OPEN                ; NO THEN HANDLE NORMALLY
 2C B441    53   90  058F  1435 159$:   MOVB    R3,@TTY$L_RB_LIN(R4)[R1]; INSERT CHARACTER
        3C  A4   B6  0594  1436          INCW    TTY$W_RB_TXTOFF(R4)     ; ADJUST POINTER
        30  A4   B6  0597  1437          INCW    TTY$W_RB_LINOFF(R4)     ; ADD TO COUNT
                        059A  1438 ;
                        059A  1439 ; CHECK FOR FULL BUFFER
                        059A  1440 ;
                        059A  1441 CHECK_BUFFER:
40  A4      3C  A4   B1  059A  1442 210$:   CMPW    TTY$W_RB_TXTOFF(R4),TTY$W_RB_TXTSIZ(R4); TEST BUFFER FULL
              13   1E  059F  1443          BGEQU   212$                    ; BUFFER FULL THEN END
                        05A1  1444 ;
                        05A1  1445 ; ECHO THE CHARACTER IF NECESSARY
                        05A1  1446 ;
                        05A1  1447 230$:   IF_STATE        <NOECHO,ESC,EDITREAD>,250$; IF NOECHO OR ESC THEN BRANCH
            FBDD    31  05AE  1448          BRW     FORMAT_CHAR             ;
            FA51    31  05B1  1449 240$:   BRW     GETNEXTCHAR             ; VECTOR TO TRANSFER MORE CHARACTERS
                        05B4  1450 ;
                        05B4  1451 ; BUFFER OVERFLOWED THEN TERMINATE
                        05B4  1452 ;
    51  38 A5   D0  05B4  1453 212$:   MOVL    UCB$L_IRP(R5),R1        ; GET THE IRP ADDRESS
        38  A1   B4  05B8  1454          CLRW    IRP$L_MEDIA(R1)         ; SET NO TERMINATOR
      1000  8F   AA  05BB  1455          BICW    #IO$M_TRMNOECHO,IRP$W_FUNC(R1); FORCE POSSIBLE ECHO
        20  A1       05BF
              18   11  05C1  1456          BRB     221$                    ; AND CONTINUE
```

```
                        05C3  1457 ;
                        05C3  1458 ; TERMINATOR FOUND BUT BUFFER NOT FULL
                        05C3  1459 ;
                        05C3  1460 TERMFOUND:
   51    58 A5    D0  05C3  1461          MOVL    UCB$L_IRP(R5),R1        ; GET THE IRP ADDRESS
   38 A1    53    B0  05C7  1462          MOVW    R3,IRP$L_MEDIA(R1)      ; SAVE LAST CHARACTER FOR STATUS
      3A A1    B6  05CB  1463          INCW    IRP$L_MEDIA+2(R1)       ; SET TERMINATOR SIZE
   51    3C A4    3C  05CE  1464          MOVZWL  TTY$W_RB_TXTOFF(R4),R1  ; GET THE OFFSET TO THE LAST CHARACTER
 00 B441    53    90  05D2  1465          MOVB    R3,@TTY$[_RB_TXT(R4)[R1]; PUT THE LAST CHARACTER AWAY
   51    58 A5    D0  05D7  1466          MOVL    UCB$L_IRP(R5),R1        ; GET THE IRP ADDRESS BACK
                        05DB  1467 221$:   SET_STATE <EOL>                 ; SET END OF LINE SEEN
CD 20 A1    0C    E0  05DF  1468 225$:   BBS     #IO$V_TRMNOECHO,IRP$W_FUNC(R1),240$; BR IF NOT TERM NOECHO
         FFBA    31  05E4  1469          BRW     230$
                        05E7  1470 ;
                        05E7  1471 ; TEST FOR ECHO NEEDS
                        05E7  1472 ;
                        05E7  1473 250$:   IF_STATE <EDITREAD,ESC>,240$    ; CONTINUE IF ESCAPE
      00    E1  05F4  1474          BBC     #TT2$V_LOCALECHO,-
   B8 48 A5        05F6  1475                  UCB$L_DEVDEPND2(R5),240$ ; NO FORMATTING IF NOT LOCAL ECHO
   51    58 A5    D0  05F9  1476          MOVL    UCB$L_IRP(R5),R1        ; GET THE IRP ADDRESS
AF 20 A1    06    E0  05FD  1477          BBS     #IO$V_NOECHO,-
                        0602  1478                  IRP$W_FUNC(R1),240$     ; OR READ NOECHO
         FBE9    31  0602  1479          BRW     FORMAT_LOCAL            ; OTHERWISE, LOCAL ECHO TERMINAL, FORMAT
```

```
                        0605  1481          .SBTTL  MOVEREADATA SERVICE ROUTINES
                        0605  1482 ;
                        0605  1483 ; TYPEAHEAD BUFFER EXHAUSTED
                        0605  1484 ;
                        0605  1485 25$:     IF_NOT_STATE NOECHO,21$          ; NO ECHO READ NECESSARY TO OPTOMIZE
                        0609  1486          SET_STATE PRE                    ; SET THE BYPASS TYPEAHED BUFFER STATE
                  0D  11  060D  1487          BRB     21$                      ; CONTINUE IN THE NORMAL CODE PATH
                        060F  1488 ;
                0027  31  060F  1489 20$:     BRW     XON                      ; CONTINUE
                        0612  1490 BUFEMPTY:
              0C A4  B4  0612  1491          CLRW    TTY$W_TA_INAHD(R4)       ; FIX THE COUNT TO REFLECT 0
                        0615  1492          IF_STATE <PASALL,NOFLTR>,25$     ;
                        061C  1493 21$:     IF_STATE <TYPFUL>,20$            ; BR IF TYPEAHEAD FULL
                        0620  1494 NOTA:
                        0620  1495 ;
                        0620  1496 ; THE TYPEAHEAD BUFFER HAS NOW BEEN EMPTIED INTO THE SYSTEM BUFFER.
                        0620  1497 ; IF PROCESSING A READ WITH ZERO SECOND TIMEOUT, RETURN THE DATA
                        0620  1498 ; TO THE USER IMMEDIATELY.
                        0620  1499 ;
        3C 68 A5  01  E1  0620  1500          BBC     #UCB$V_TT_TIMO,UCB$W_DEVSTS(R5),DISMISS; BR IF NOT READ WITH TIMEOUT
          00000000'EF  D1  0625  1501          CMPL    TTY$A_MAXTIME,UCB$L_TT_RDUE(R5); ZERO SECOND TIMEOUT?
              00B0 C5       062B
                   31  12  062E  1502          BNEQ    DISMISS                  ; BR IF NOT.
                0220 8F  B0  0630  1503          MOVW    #SS$_TIMEOUT,UCB$W_BOFF(R5); SET TIMEOUT COMPLETION STATUS.
                  7C A5       0634
;MIR0001                F9C7' 31  0636    .1          BRW     TTY$READONE              ; COMPLETE REQUEST.
-1                              0639  1505
```

```
                     0639  1507 ;++
                     0639  1508 ; XON - SEND XON
                     0639  1509 ;
                     0639  1510 ; FUNCTIONAL DESCRIPTION:
                     0639  1511 ;
                     0639  1512 ; THIS ROUTINE IS USED TO SEND AN XON
                     0639  1513 ;
                     0639  1514 ; INPUTS:
                     0639  1515 ;
                     0639  1516 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                     0639  1517 ;       R5 = UCB ADDRESS
                     0639  1518 ;
                     0639  1519 ; OUTPUTS:
                     0639  1520 ;
                     0639  1521 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                     0639  1522 ;       R3 = XON
                     0639  1523 ;       R5 = UCB ADDRESS
                     0639  1524 ;--
                     0639  1525 XON:                                    ; SEND AN XON
                     0639  1526         CLR_STATE <TYPFUL>
                     063D  1527         IF_NOT_STATE <OVRFLO>,10$       ; SKIP IF NO OVERFLOW ERROR
                     0641  1528
                     0641  1529         CLR_STATE <OVRFLO>              ; RESET CONDITION
0838 8F    B0       0645  1530         MOVW    #SS$_DATAOVERUN,UCB$W_BOFF(R5); COMPLETE READ IN ERROR
  7C A5
   F9B2'   30       0649
                     064B  1531         BSBW    TTY$READONE
                     064E  1532
   F9AF'   30       064E  1533 10$:    BSBW    TTY$XON
   F9B1    31       0651  1534         BRW     GETNEXTCHAR
```

```
                     0654  1536
                     0654  1537 ;
                     0654  1538 ; CONVERT CHARACTER TO UPPER CASE
                     0654  1539 ;
0000'CF43   03   E1  0654  1540 CVTLOW: BBC     #TTY$V_CH_LOWER,W^TTY$A_TYPE[R3],154$; BR IF NOT LOWER CHARACTER
                 03  065A
      53   20   AA  065B  1541         BICW    #^X020,R3               ; CONVERT TO UPPER CASE
         FF20   31  065E  1542 154$:   BRW     155$
```

```
                     0661  1544 ;
                     0661  1545 ; DISMISS INTERRUPT
                     0661  1546 ;
                     0661  1547 DISMISS:
010B C5   94         0661  1548          CLRB    UCB$B_TT_OUTYPE(R5)     ; SET NO RETURN CHARACTER.
          05         0665  1549          RSB                             ; and dismiss the interupt
```

```
                                0666  1551 ;
                                0666  1552 ; ESCAPE SEQUENCE IN PROGRESS - CHECK SYNTAX
                                0666  1553 ;
                                0666  1554 ESCINPROG:
                     53   95    0666  1555          TSTB    R3                          ; NO NULLS IN ESCAPE SEQUENCES
                     50   13    0668  1556          BEQL    961$                        ; SO DROP IT
                  51   3C A4    066A  1557          MOVZWL  TTY$W_RB_TXTOFF(R4),R1      ; GET THE PLACE TO STORE THE ESCAPE
               00 B441    53   066E  1558          MOVB    R3,@TTY$C_RB_TXT(R4)[R1]    ; AND PLACE THE CHARACTER IN THE RIGHT PLACE
                  3C A4    B6   0673  1559          INCW    TTY$W_RB_TXTOFF(R4)         ; AND INCREMENT THE WORLD
                  51 58 A5 D0   0676  1560          MOVL    UCB$L_IRP(R5),R1            ; GET THE IRP ADDRESS
                     3A A1 B6   067A  1561          INCW    IRP$L_MEDIA+2(R1)           ; ADJUST SIZE OF ESCAPE SEQUENCE
                      FE34 30   067D  1562          BSBW    ESCSYNTAX                   ; CHECK CHAR AGAINST SYNTAX
                        31 14   0680  1563          BGTR    960$                        ; VALID CHAR, CONTINUE INPUT
                        04 13   0682  1564          BEQL    197$                        ; VALID CHAR, SEQUENCE COMPLETE
                                0684  1565 ;
                                0684  1566 ; ESCAPE SYNTAX ERROR
                                0684  1567 ;
                                0684  1568 195$:    SET_STATE BADESC                    ; SET BAD ESCAPE
                                0688  1569 197$:    CLR_STATE ESC                       ; SET SEQUENCE DONE
                  3C A4   3A A1 A2  068D  1570          SUBW    IRP$L_MEDIA+2(R1),TTY$W_RB_TXTOFF(R4); ADJUST TRANSFER SIZE
                                0692  1571          IF_STATE <NOFLTR,BADESC>,199$       ; BAD SEQUENCE THEN HANDLE NORMALY OR
                                069A  1572                                              ; NOFILTER READ
:MIR0001                        069A    1          IF_NOT_STATE <EDITING>,199$         ; NO EDITING THE GO AND RETURN THE SEQUENDE
                     53   7E 8F   069E  1573          CMPB    #^A/~/,R3                   ; IS THIS A FUNCTION KEY
                        4C 12   06A2  1574          BNEQ    910$                        ; NO THEN HANDLE APROPRIATELY
                     48 A4   15 B1   06A4  1575          CMPW    #21,TTY$W_RB_ESCTKN(R4)     ; IS THIS THE EXIT KEY
                        13 12   06A8  1576          BNEQ    920$                        ; NO THEN TRY NORMAL KEY
                        53   1A 9A   06AA  1577          MOVZBL  #TTY$C_CTRLZ,R3             ; SETUP A CONTROL-Z
                           38 A1 D4   06AD  1578          CLRL    IRP$L_MEDIA(R1)             ; CLEAN THE ESCAPE SEQUENCE UP
                            018B 31   06B0  1579          BRW     162$                        ;AND GO OFF TO DISPATCH
                                06B3  1580
                            0078 31   06B3  1581 960$:    BRW     198$
                                06B6  1582
                                06B6  1583 199$:
                                06B6  1584          SET_STATE EOL                       ; SET UP END OF LINE
                            F948 31   06BA  1585 961$:    BRW     GETNEXTCHAR                 ; THEN GET THE NEXT CHARACTER
                                06BD  1586
                                06BD  1587 ;
                                06BD  1588 ; FUNCTION KEY DETECTED CHANGE FUNCTION KEY TO TOKEN
                                06BD  1589 ; BUT ONLY IF WE ARE EDITING
                                06BD  1590 ;
                                06BD  1591 920$:    IF_NOT_STATE <EDITING>,199$         ; NO EDITING THE GO AND RETURN THE SEQUENDE
                        48 A4   B1 06C1  1592          CMPW    TTY$W_RB_ESCTKN(R4),#TTY$K_MAXESCTKN; IS IT OUT OF RANGE
                     0000'8F         06C4
                        57   1A 06C7  1593          BGTRU   190$                        ; YES THEN RETURN IT
                  51    48 A4   3C 06C9  1594          MOVZWL  TTY$W_RB_ESCTKN(R4),R1      ; GET THE OFFSET
               00000000'EF41    9A 06CD  1595          MOVZBL  TTY$A_FCNTKN[R1],R1         ; AND MAKE THAT INTO A TOKEN
                        51      06D4
                                06D5  1596 930$:
                        51   95 06D5  1597          TSTB    R1                          ; IS THIS VALID
                        47   13 06D7  1598          BEQL    190$                        ; NO THEN RETURN THE SEQUENCE
                  53    58 A5   D0 06D9  1599 191$:    MOVL    UCB$L_IRP(R5),R3            ; GET THE IRP ADDRESS
                     38 A3   D4 06DD  1600          CLRL    IRP$L_MEDIA(R3)             ; CLEAR THE ESCAPE SEQUENCE OUT
                     0D   51 91 06E0  1601          CMPB    R1,#TTY$K_ET_UNUSED         ; IS THIS AN UNUSED ESCAPE SEQUENCE
                        3F   13 06E3  1602          BEQL    970$                        ; YES THEN IGNORE IT...
                     0B   51 91 06E5  1603          CMPB    R1,#TTY$K_ET_RECALL         ; IS THIS A RECALL COMMAND
                        03   12 06E8  1604          BNEQ    192$                        ; NO THEN CONTINUE NORMALY
```

```
                53   02   9A  06EA  1605            MOVZBL  #TTY$C_CTRLB,R3          ; YES THEN MAKE IT A CONTROL-B AND CONTINUE
                  0127   31  06ED  1606 192$:        BRW     70$
                         06F0  1607
                         06F0  1608 910$:        IF_NOT_STATE <EDITING>,199$     ; AND WE ARE NOT EDITING DO THE SAME
             03    3A A1   B1  06F4  1609            CMPW    IRP$L_MEDIA+2(R1),#3    ; IS THE SEQUENCE LONGER THAN 3 CHARACTERS
                     26   14  06F8  1610            BGTR    190$                    ; YES THEN DO A MATCH ON IT
                51   05   9A  06FA  1611            MOVZBL  #TTY$K_ET_BACK_CHAR,R1  ; SETUP FOR BACK CHARCTER
                53    44 8F   91  06FD  1612            CMPB    #^A/D/,R3               ; IS IT A BACK CHARACTER
                     D2   13  0701  1613            BEQL    930$                    ; YES THEN DISPATCH
                51   06   9A  0703  1614            MOVZBL  #TTY$K_ET_FORWARD_CHAR,R1; GO FOWARD A CHARACTER
                53    43 8F   91  0706  1615            CMPB    #^A/C/,R3               ; IS IT A FORWARD
                     C9   13  070A  1616            BEQL    930$                    ; HANDLE IT
                51   0B   9A  070C  1617            MOVZBL  #TTY$K_ET_RECALL,R1     ; RECALL A COMMAND
                53    41 8F   91  070F  1618            CMPB    #^A/A/,R3               ; IS THIS RECALL?
                     C0   13  0713  1619            BEQL    930$
          06 20 A4   10   E0  0715  1620            BBS     #TRM$V_TM_NORECALL,TTY$L_RB_MOD(R4),190$; CHECK NORECALL
                53    42 8F   91  071A  1621            CMPB    #^A/B/,R3               ; IS THIS A DOWN ARROW?
                     B5   13  071E  1622            BEQL    930$                    ; YES THEN DROP IT
                         0720  1623 ;
                         0720  1624 ; DON'T TERMINATE ON THIS SEQUENCE UNLESS HE REALY WANTS IT
                         0720  1625 ; (THE MODE OF THE READ IS ESCAPE
                         0720  1626 ;
                         0720  1627 190$:        IF_STATE ESCAPE,199$            ; IF THIS IS AN ESCAPE SEQUENCE READ THEN
                         0724  1628 970$:                                        ; TERMINATE ON ESCAPE SEQUENCES.
                51   58 A5   D0  0724  1629            MOVL    UCB$L_IRP(R5),R1        ; MAKE SURE WE HAVE THE IRP ADDRESS
                     38 A1   D4  0728  1630            CLRL    IRP$L_MEDIA(R1)         ; NO THEN CLEAN THE SEQUENCE OFF THE TERMINA
                      F8D7   31  072B  1631            BRW     GETNEXTCHAR             ; AND GET THE NEXT CHARACTER
                         072E  1632 ;
                         072E  1633 ; NORMAL ESCAPE SEQUENCE TERMINATION
                         072E  1634 ;
                         072E  1635
                53    30   91  072E  1636 198$:        CMPB    #^A/0/,R3               ; IS THIS A NUMBER
                     10   14  0731  1637            BGTR    900$                    ; NO THEN EXIT
                53    39   91  0733  1638            CMPB    #^A/9/,R3               ; ...
                     0B   19  0736  1639            BLSS    900$                    ;
          48 A4   0A   A4  0738  1640            MULW    #10,TTY$W_RB_ESCTKN(R4) ; SHIFT THE NUMBER OVER
                53   30   A2  073C  1641            SUBW    #^A/0/,R3               ; SUBTRACT OUT 0
          48 A4   53   A0  073F  1642            ADDW    R3,TTY$W_RB_ESCTKN(R4)  ;ADD IT IN
                FE54   31  0743  1643 900$:        BRW     210$
```

```
                     0746  1645 ;
                     0746  1646 ; SEE IF DELETE SEQUENCE SHOULD BE TERMINATED
                     0746  1647 ;
7F 8F    53    91    0746  1648 INDELST:CMPB    R3,#TTY$C_DELETE        ; IS THIS CHARACTER A DELETE?
         0E    13    074A  1649          BEQL    50$                     ; IF YES THEN GO ON
      0C A4    B6    074C  1650          INCW    TTY$W_TA_INAHD(R4)      ; NO THEN PUT THE CHARACTER BACK
                     074F  1651          CLR_STATE DEL                   ; TERMINATE DELETE SEQUENCE
                     0753  1652 ;
                     0753  1653 ; TERMINATE DELETE SEQUENCE IF NOT IN NO-ECHO
                     0753  1654 ;
53   5C 8F    9A    0753  1655          MOVZBL  #^A/\/,R3               ; SET OUTPUT CHARACTER
       FA34   31    0757  1656          BRW     FORMAT_CHAR             ; FORMAT THE CHARACTER
                     075A  1657
       FDF2   31    075A  1658 50$:     BRW     40$
                     075D  1659
```

```
                     075D  1661 ;
                     075D  1662 ; OPEN THE BUFFER
                     075D  1663 ; OR JUST OVERSTRIKE
                     075D  1664 ;
                     075D  1665 OPEN:
                     075D  1666 ;
                     075D  1667 ; SPREAD THE BUFFER TO PUT THIS CHARCTER IN PLACE
                     075D  1668 ;
          3F   BB    075D  1669          PUSHR   #^M<R0,R1,R2,R3,R4,R5>  ; SAVE THE REGISTERS OVER THE MOVE
          51   D4    075F  1670          CLRL    R1                      ; SET FORWARD MODE
        07F9   30    0761  1671          BSBW    CHAR_SIZE               ; GET THE CHARACTER SIZE
                     0764  1672 ;
                     0764  1673 ; IF NOT AT THE END OF LINE THEN DON'T LET THE USER WRAP THE LINE
                     0764  1674 ;
                     0764  1675          IF_STATE <WRAP,TABRIGHT>,194$           ;
00FC C5    53    A1  076E  1676 169$:    ADDW3   R3,UCB$W_TT_CURSOR(R5),TTY$W_RB_CPZCUR(R4); AND ADD IT INTO THE CURR
        38 A4        0773
   51    30 A4  3C   0775  1677          MOVZWL  TTY$W_RB_LINOFF(R4),R1  ; GET THE LOCATION OF THE LAST CHARACTER
      2C B441   9E   0779  1678          MOVAB   @TTY$L_RB_LIN(R4)[R1],TTY$L_RB_ECHSTR(R4); GET THE ADDRESS
        14 A4        077D
   44 A4    01   B0   077F  1679          MOVW    #TTY$K_ER_CLRECHO,TTY$W_RB_MODE(R4); AND SETUP THE ECHO MODE
                     0783  1680          IF_STATE NOECHO,193$            ; DON'T ECHO IF NOT ECHOING...
                     0787  1681          SET_STATE EDITREAD              ; AND SET THE EDIT READ STATE
                     078B  1682 193$:    IF_NOT_STATE OVERSTRIKE,158$    ; NOT IN OVERSTRIKE THEN HANDLE NORMALY
09    0C AE    91   078F  1683          CMPB    12(SP),#TTY$C_TAB       ; WAS THIS CHARACTER A TAB?
          2D    13   0793  1684          BEQL    158$                    ; YES THEN INSERT IT.
                     0795  1685
        32 A4    B7   0795  1686          DECW    TTY$W_RB_LINREST(R4)    ; IN OVERSTRIKE WE DELETE WHILE TYPING
        3C A4    B7   0798  1687          DECW    TTY$W_RB_TXTOFF(R4)     ; FAKE OUT THE REST OF THE LOGIC
                     079B  1688
      01    53    D1   079B  1689          CMPL    R3,#1                   ; IS THIS A SHORT ECHOING CHARACTER
          2C    12   079E  1690          BNEQ    168$                    ; NO THEN OUTPUT THE WHOLE THING
09    14 B4    91   07A0  1691          CMPB    @TTY$L_RB_ECHSTR(R4),#TTY$C_TAB; ARE WE ON TOP OF A TAB?
          0C    13   07A4  1692          BEQL    157$                    ; YES THEN HANDLE SPECAILLY
        44 A4    B4   07A6  1693          CLRW    TTY$W_RB_MODE(R4)       ; CLEAN OUT THE MODE FOR LATER
        14 A4    D4   07A9  1694          CLRL    TTY$L_RB_ECHSTR(R4)     ; MAKE SURE THE STRING IS ZERO
                     07AC  1695          CLR_STATE EDITREAD              ; AND DON'T DO THE EDITREAD ECHO (ONLY
                     07B0  1696                                          ; ONE CHARACTER NO NEED)
          1A   11    07B0  1697          BRB     168$                    ; AND DON'T BOTHER WITH THE MOVC
   03     00   EF    07B2  1698 157$:    EXTZV   #0,#3,TTY$W_RB_CPZCUR(R4),R3; GET THE LENGTH OF THE TAB
53    38 A4        07B5
          53    D5   07B8  1699          TSTL    R3                      ; DELETE THE TAB WHEN IT EXPANDS TO ZERO
          1B    13   07BA  1700          BEQL    171$                    ; MAKE IT DISAPPEAR
        32 A4    B6   07BC  1701          INCW    TTY$W_RB_LINREST(R4)    ; ELSE JUST INSERT THE CHARACTER
        3C A4    B6   07BF  1702          INCW    TTY$W_RB_TXTOFF(R4)
          51    D6   07C2  1703 158$:    INCL    R1                      ; INCREMENT THE REGISTER BY 1 TO GET THE
                     07C4  1704                                          ; NEW POSITION
        32 A4    28   07C4  1705          MOVC3   TTY$W_RB_LINREST(R4),-  ; AND MOVE THE CHARACTERS OVER
        14 B4         07C7  1706                  @TTY$L_RB_ECHSTR(R4),@TTY$L_RB_LIN(R4)[R1]
   2C B441        07C9
          3F    BA   07CC  1707 168$:    POPR    #^M<R0,R1,R2,R3,R4,R5>  ; RESTORE THE REGISTERS
01      32 A4    81   07CE  1708          ADDB3   TTY$W_RB_LINREST(R4),#1,TTY$B_RB_ECHLEN(R4);AND LENGTH OF STRING TO
        08 A4        07D2
         FDB8   31    07D4  1709          BRW     159$                    ; AND INSERT THE CHARACTER
        30 A4    B6   07D7  1710 171$:    INCW    TTY$W_RB_LINOFF(R4)     ; LOOK FROM THIS CHARACTER ON
         08BB   30    07DA  1711          BSBW    TABRIGHT                ; MAKE SURE THE TAB AFTER THIS POSITION
                     07DD  1712                                          ; FLAG IS SET CORRECTLY.
```

```
                30 A4   B7  07DD  1713          DECW    TTY$W_RB_LINOFF(R4)     ; KEEP LINOFF THE SAME AS WE STARTED WITH
                   EA   11  07E0  1714          BRB     168$                    ; JOIN THE FLOW
                        07E2  1715 ;
                        07E2  1716 ; ONCE WRAP IS DETERMINED THEN CHECK SPECIFICALLY IF THIS CHARACTER IS VALID
                        07E2  1717 ;
                        07E2  1718 194$:    IF_NOT_STATE OVERSTRIKE,234$    ; DON'T EVER INSERT AFTER THE WRAP
             01    53   B1  07E6  1719          CMPW    R3,#1                   ; IS THIS A SINGLE CHARACTER EXPANSION
                   06   12  07E9  1720          BNEQ    234$                    ; NO THEN DON'T INSERT IT
          09    0C AE   91  07EB  1721          CMPB    12(SP),#TTY$C_TAB       ; WAS THIS CHARACTER A TAB?
                   05   12  07EF  1722          BNEQ    235$                    ; NO THEN OVERSTRIKE IT
                   3F   BA  07F1  1723 234$:    POPR    #^M<R0,R1,R2,R3,R4,R5>  ; RESTORE THE REGISTERS
                 F80F   31  07F3  1724          BRW     GETNEXTCHAR             ; AND GET THE NEXT CHARACTER
                 FF75   31  07F6  1725 235$:    BRW     169$
```

```
                                                M 14
                        07F9  1727 ;
                        07F9  1728 ; QUOTING  CHARACTER DETECTED
                        07F9  1729 ;
 OD 04 A2    14   E4   07F9  1730 QUOTE:  BBSC    #TTY$V_ST_EDITING,4(R2),65$; IF EDITING IS SET THEN CLEAR IT
                        07FE  1731          IF_STATE        ESC,65$          ; IF ESCAPE THEN DON'T TURN OFF EDITING YET
                        0802  1732          SET_STATE EDITING                ; IF IT NOT EDITING THEN SET EDITING
                        0806  1733          CLR_STATE QUOTING                ; AND CLEAR QUOTING
              FD5C  31  080B  1734 65$:     BRW     67$                      ; RETURN TO THE NORMAL PLACE
```

```
                     080E  1736          .SBTTL  Specail input character dispatcher
                     080E  1737 ;++
                     080E  1738 ;
                     080E  1739 ; SERVICE ROUTINES TO MOVE READATA
                     080E  1740 ;
                     080E  1741 ; SPECAIL EDITING CHARACTER HANDELING
                     080E  1742 ;
                     080E  1743 EDITINGCHAR:
                     080E  1744          IF_STATE <EDITING>,70$              ; EDITING THEN NO RANGE CHECK NECESSARY
          04 51 D1   0812  1745          CMPL    R1,#TTY$K_EDITNORMAL        ; NOT EDITING THEN ANYTHING ADVANCED
             27 14   0815  1746          BGTR    162$                        ; IS BAD
                     0817  1747 70$:     CASE    R1,TYPE=L,<-
                     0817  1748                  151$,-                      ; IS NORMAL CHARACTER (SHOULD NEVER BE USED)
                     0817  1749                  CTRLU,-                     ; IS CONTROL-U
                     0817  1750                  CTRLR,-                     ; INDICATES CONTROL-R
                     0817  1751                  DELCHAR,-                   ; INDICATES DELETE CHARCTER
                     0817  1752                  ESCAPE_CHAR,-               ; INDICATES ESCAPE CHARCTER
                     0817  1753                  BACK_CHAR,-                 ; MOVE BACK ONE CHARACTER
                     0817  1754                  FORWARD_CHAR,-              ; MOVE FOWARD ONE CHARACTER
                     0817  1755                  MOVE_EOL,-                  ; MOVE TO THE END OF LINE
                     0817  1756                  MOVE_BOL,-                  ; MOVE TO THE BEGINNING OF LINE
                     0817  1757                  DELETE_WORD,-               ; DELETE WORD TO THE LEFT
                     0817  1758                  QUOTING,-                   ; QUOTE CHARACTER
                     0817  1759                  RECALL,-                    ; RECALL THE LAST COMMAND
                     0817  1760                  TOGGELINSOV-                ; TOGGEL INSERT/OVERSTRIKE MODE
                     0817  1761                  >
                     0835  1762          IF_STATE TERMNORM,162$              ; TERMINATE ON USUAL
                     0839  1763                                              ; CHARACTERS
          0E 51 D1   0839  1764          CMPL    R1,#TTY$K_ET_TERMINATE      ; WAS THIS A REAL TERMINATOR CHARACTER
             1E 12   083C  1765          BNEQ    165$                        ; NO THEN DON'T LET IT THRU
                     083E  1766 162$:    IF_STATE        <ESC>,166$          ; IF IN AN ESCAPE SEQUENCE THEN LET IT THRU
OF 1C B4    53 E0   0842  1767          BBS     R3,@TTY$L_RB_TERM(R4),164$; IS THIS CHARACTER A TERMINATOR
          0D 53 91   0847  1768          CMPB    R3,#TTY$C_CR                ; WE ARE PREPARED TO HANDLE <CR>
             13 13   084A  1769          BEQL    800$                        ; SO LET IT THRU ANYWAY
                     084C  1770          IF_STATE <QUOTING,EDITING>,165$ ; NO THEN DROP IT IF WE ARE EDITING
           FD1E 31   0853  1771 167$:    BRW     151$
           FD6A 31   0856  1772 164$:    BRW     TERMFOUND                   ; TERMINATOR DETECTED
           FE0A 31   0859  1773 166$:    BRW     ESCINPROG                   ; ESCAPE SEQUENCE INPROGRESS
           F7A6 31   085C  1774 165$:    BRW     GETNEXTCHAR                 ; GET THE NEXT CHAR
         32 A4 B5    085F  1775 800$:    TSTW    TTY$W_RB_LINREST(R4)        ; ARE WE AT THE END OF THE LINE
             EF 13   0862  1776          BEQL    167$                        ; YES THEN CR IS LEGAL
           F79E 31   0864  1777          BRW     GETNEXTCHAR
                     0867  1778
                     0867  1779          .disable lsb
                     0867  1780
```

```
                     0867  1782 ;++
                     0867  1783 ; RECALLING
                     0867  1784 ;
                     0867  1785 ; DESCRIPTION:
                     0867  1786 ;       THIS STATE IS USED WHEN THE LAST COMMAND TYPED IS REQUESTED.
                     0867  1787 ; THIS ROUTINE WILL REMOVE CHARACTERS FROM THE RECALL BUFFER THEN
                     0867  1788 ; ALLOW THE NORMAL INSERT CHARACTER LOGIC TAKE AFFECT.  THE CHARACTERS
                     0867  1789 ; IN THE RECALL BUFFER ARE PROCESSED AS IF THEY WERE NO FORMAT.
                     0867  1790 ;
                     0867  1791 ; INPUTS:
                     0867  1792 ;       R5 = UCB ADDRESS
                     0867  1793 ;
                     0867  1794 ; OUTPUTS:
                     0867  1795 ;       SEE MOVEREADATA
                     0867  1796 ;--
                     0867  1797 RECALLING:
 51   00E4 C5   D0   0867  1798         MOVL    UCB$L_TT_TYPAHD(R5),R1  ; GET THE ADDRESS OF THE TYPEAHEAD BUFFER
            2C   13   086C  1799         BEQL    20$                     ; NONE THEN LET MOVEREADATA HANDLE IT
   54    78 A5   D0   086E  1800         MOVL    UCB$L_SVAPTE(R5),R4     ; SETUP THE READ PACKET ADDRESS
   53    0E A1   3C   0872  1801         MOVZWL  TTY$W_TA_RCLOFF(R1),R3  ; GET THE ADDRESS OF THE NEXT CARACTER
 53    18 A143   9A   0876  1802         MOVZBL  TTY$A_TA_RCL(R1)[R3],R3 ; GET THE CHARACTER
    0E A1   B6   087B  1803         INCW    TTY$W_TA_RCLOFF(R1)     ; MOVE ON TO THE NEXT CHARACTER
14 A1    0E A1   B1   087E  1804         CMPW    TTY$W_TA_RCLOFF(R1),TTY$W_TA_RCLSIZ(R1); ARE WE FINISHED
            07   18   0883  1805         BGEQ    5$                      ; YES THEN TURN OFF RECALL AND CONTINUE
40 A4    0E A1   B1   0885  1806         CMPW    TTY$W_TA_RCLOFF(R1),TTY$W_RB_TXTSIZ(R4); ARE WE GOING TO OVERFLOW
            04   19   088A  1807         BLSS    10$                     ; THE USERS BUFFER
                      088C  1808 5$:     CLR_STATE RECALL                ; YES THEN MAKE THIS THE LAST TIME
          FCE5   31   0890  1809 10$:    BRW     INSERT_CHAR             ; AND INSERT THE CHARACTER
          FC97   31   0893  1810 20$:    BRW     MOVEREADATA             ; ABORT EXIT
```

```
                     0896  1812          .SBTTL  Post typeahead character action routines
                     0896  1813
                     0896  1814 ;++
                     0896  1815 ; BACK_CHAR
                     0896  1816 ;       Move backward a single character.
                     0896  1817 ;--
                     0896  1818 BACK_CHAR:
          3A A4  B5  0896  1819          TSTW    TTY$W_RB_CPZORG(R4)      ; START AT ZERO?
             03  13  0899  1820          BEQL    10$                      ; YES THEN NO SPECAIL ACTION
           0885  30  089B  1821          BSBW    ZERO_CPZORG              ; NO THEN MAKE SURE WE DO
          30 A4  B7  089E  1822 10$:     DECW    TTY$W_RB_LINOFF(R4)      ; BACKUP A CHARACTER
             2A  19  08A1  1823          BLSS    50$                      ; AT THE END THEN DON'T ALLOW IT
          32 A4  B6  08A3  1824          INCW    TTY$W_RB_LINREST(R4)     ; UPDATE THE REST POINTER
       51 30 A4  3C  08A6  1825          MOVZWL  TTY$W_RB_LINOFF(R4),R1   ; GET THE OFFSET TO THE CHARACTER
    53 2C B441   9A  08AA  1826          MOVZBL  @TTY$C_RB_LIN(R4)[R1],R3 ; GET THE CHARCTER
          53 09  91  08AF  1827          CMPB    #TTY$C_TAB,R3            ; IS THIS A TAB
             05  12  08B2  1828          BNEQ    20$                      ; NO THEN CONTINUE
                  08B4  1829          SET_STATE TABRIGHT                 ; ELSE MARK THIS INFORMATION
          51 01  CE  08B9  1830 20$:     MNEGL   #1,R1                    ; GOING BACKWARDS TO
           069E  30  08BC  1831          BSBW    CHAR_SIZE                ; CHECK THE SIZE OF THE CHARACTER
00FC C5    53  A3  08BF  1832          SUBW3   R3,UCB$W_TT_CURSOR(R5),TTY$W_RB_CPZCUR(R4); UPDATE THE CURSOR POSITI
          38 A4        08C4
                  08C6  1833          IF_STATE NOECHO,60$
           07E7  31  08CA  1834          BRW     UPDATE_CURSOR            ; SET THE CURSOR TO IT'S NEW LOCATION
                  08CD  1835
          30 A4  B4  08CD  1836 50$:     CLRW    TTY$W_RB_LINOFF(R4)      ; RESET THE POINTER
           F9CD  31  08D0  1837 60$:     BRW     DROP                     ; AND DROP THE CHARACTER
```

```
                                                D 15
                     08D3  1839          .enable lsb
                     08D3  1840 ;
                     08D3  1841 ; CONTROL R PROCESSING
                     08D3  1842 ;
                     08D3  1843 CTRLR:   IF_STATE <ESC>,150$                 ; NO ACTION IF ESCAPE OR NO ECHO
                     08D7  1844          IF_NOT_STATE NOECHO,100$            ; BRANCH IF ECHO
           00   E1   08DB  1845          BBC     #TT2$V_LOCALECHO,-
      2D 48 A5        08DD  1846                  UCB$L_DEVDEPND2(R5),150$; BRANCH IF NOT LOCAL ECHO
                     08E0  1847
    54  0000'CF   9E   08E0  1848 100$:   MOVAB   W^TTY$A_CTRLR,R4            ; ADDRESS MULTIECHO STRING
           0C   E1   08E5  1849 110$:   BBC     #TT2$V_EDITING,-
      1C 48 A5        08E7  1850                  UCB$L_DEVDEPND2(R5),130$        ; NO THEN USE NORMAL ECHO STRING
17 44 A5   0C   E1   08EA  1851          BBC     #TT$V_SCOPE,UCB$L_DEVDEPEND(R5),130$; DON'T SKIP STUFF
                     08EF  1852                                              ; ON A HARD COPY TERMINAL
    51   78 A5   D0   08EF  1853          MOVL    UCB$L_SVAPTE(R5),R1         ; GET THE PACKET ADDRESS
         3A A1   B5   08F3  1854          TSTW    TTY$W_RB_CPZORG(R1)         ; WERE WE AT THE FIRST COLUMN WHEN
           04    12   08F6  1855          BNEQ    120$                        ; WE STARTED THE READ
                     08F8  1856          SET_STATE <SKIPCRLF>                ; NO LINEFEEDS AFTER <CR>
           18   E1   08FC  1857 120$:    BBC     #TT2$V_ANSICRT,-            ; IS IT AN ANSI CRT
      05 48 A5        08FE  1858                  UCB$L_DEVDEPND2(R5),130$; NO THEN USE NORMAL ECHO STRING
    54  0000'CF   9E   0901  1859          MOVAB   W^TTY$A_ANSI_DEOL,R4        ; YES THEN WE CAN USE CRT MODE
                     0906  1860 130$:
                     0906  1861          SET_STATE       <EDITREAD>          ; SET CONTROL R STATE
         075E    31   090A  1862          BRW     STRTMULTI                   ; START MULTIPLE OUTPUT SEQUENCE
                     090D  1863
         FC1D    31   090D  1864 150$:    BRW     MOVEREADATA
```

```
                        0910  1866 ;
                        0910  1867 ; CONTROL U PROCESSING
                        0910  1868 ;
                 0239   30   0910  1869 10$:    BSBW    DELESCAPE                       ; DELETE THE ESCAPE SEQUENCE
                        0913  1870 CTRLU:  IF_STATE ESC,10$                            ; IF IN ESCAPE SEQUENCE THEN DELETE ESCAPE
              30 A4     B5   0917  1871          TSTW    TTY$W_RB_LINOFF(R4)           ; ANY DATA TO CONTROL-U?
                 69     13   091A  1872          BEQL    50$                           ; NO THEN DON'T DO THE WORK
3C A4         30 A4     A2   091C  1873          SUBW    TTY$W_RB_LINOFF(R4),TTY$W_RB_TXTOFF(R4) ; CLEAN OUT THE TEXT OFFSETO
              0B A4     94   0921  1874          CLRB    TTY$B_RB_ECHLEN(R4)           ; CLEAN THE EXTRA
              38 A4     B4   0924  1875          CLRW    TTY$W_RB_CPZCUR(R4)           ; MOVE TO THE BEGINNING OF THE LINE
              3A A4     B5   0927  1876          TSTW    TTY$W_RB_CPZORG(R4)           ; WAS THIS AT THE RIGHT
                 04     12   092A  1877          BNEQ    12$                           ; NO THEN USE THE OLD METHOD
           44 A4   06     9B   092C  1878          MOVZBW  #TTY$K_ER_CLRREST,TTY$W_RB_MODE(R4); EXIT WHEN NOT DOING ANYTHING EL
              32 A4     B5   0930  1879 12$:    TSTW    TTY$W_RB_LINREST(R4)          ; IS THERE A REST OF THE LINE
                 26     13   0933  1880          BEQL    13$                           ; NO THEN HANDLE NORMALLY
                 3F     BB   0935  1881          PUSHR   #^M<R0,R1,R2,R3,R4,R5>        ; SAVE THE REGISTERS
           51    30 A4     3C   0937  1882          MOVZWL  TTY$W_RB_LINOFF(R4),R1        ; GET THE NUMBER OF CHARACTERS
              32 A4     28   093B  1883          MOVC3   TTY$W_RB_LINREST(R4),@TTY$L_RB_LIN(R4)[R1],@TTY$L_RB_LIN(R4);
                 2C B441        093E
                 2C B4          0941
                 3F     BA   0943  1884          POPR    #^M<R0,R1,R2,R3,R4,R5>        ; AND RESTORE THE REGISTERS
              3A A4     B5   0945  1885          TSTW    TTY$W_RB_CPZORG(R4)           ; WAS THIS AT THE RIGHT
                 1B     12   0948  1886          BNEQ    14$                           ; NO THEN USE THE OLD METHOD
           44 A4   01     9B   094A  1887          MOVZBW  #TTY$K_ER_CLRECHO,TTY$W_RB_MODE(R4); SETUP THE NEW ECHO MODE
14 A4         2C A4     D0   094E  1888          MOVL    TTY$L_RB_LIN(R4),TTY$L_RB_ECHSTR(R4); THE LOCATION OF THE FIRST CHAR
0B A4         32 A4     90   0953  1889          MOVB    TTY$W_RB_LINREST(R4),TTY$B_RB_ECHLEN(R4); AND THE LENGTH
              38 A4     B4   0958  1890          CLRW    TTY$W_RB_CPZCUR(R4)           ; AND GO ALL THE WAY TO THE BEGINNING
           64    2C A4     D1   095B  1891 13$:    CMPL    TTY$L_RB_LIN(R4),TTY$L_RB_TXT(R4); ARE WE ON THE FIRST LINE
                 04     12   095F  1892          BNEQU   14$                           ;NO THEN CONTINUE AS USUAL
           44 A4   07     9B   0961  1893          MOVZBW  #TTY$K_ER_PRMECHO,TTY$W_RB_MODE(R4); ECHO THE PROMPT TOO
              30 A4     B4   0965  1894 14$:    CLRW    TTY$W_RB_LINOFF(R4)           ; NOW ZERO THE LINE OFFFSET
                        C968  1895
                        0968  1896          IF_STATE NOECHO,15$                       ; BRANCH IF ECHO
        54    0000'CF   9E   096C  1897          MOVAB   W^TTY$A_CTRLU,R4              ; ADDRESS MULTIECHO STRING
                 FF71   31   0971  1898          BRW     110$                          ; CONTINUE
                        0974  1899
                 FBB6   31   0974  1900 15$:    BRW     MOVEREADATA                   ;
                        0977  1901 55$:    CLR_STATE <WRAP,CURSOR>
                 FF91   31   097F  1902          BRW     CTRLU
                 FF4E   31   0982  1903 57$:    BRW     CTRLR
                        0985  1904 ;
                        0985  1905 ; NO DATA TO CONTROL-U
                        0985  1906 ;
              32 A4     B5   0985  1907 50$:    TSTW    TTY$W_RB_LINREST(R4)          ; ANY DATA AFTER THIS POINT
                 EA     12   0988  1908          BNEQ    15$                           ; YES THEN JUST RETURN
              3C A4     B5   098A  1909          TSTW    TTY$W_RB_TXTOFF(R4)           ; IS THERE ANY REASON TO DO THIS
                 E5     13   098D  1910          BEQL    15$                           ; NO THEN DON'T BOTHER
              3C A4     B7   098F  1911          DECW    TTY$W_RB_TXTOFF(R4)           ; REMOVE THE LAST CHARACTER
               0637     30   0992  1912          BSBW    FIND_BOL                      ; THEN FIND THE BEGINNING
                        0995  1913                                                    ; OF THE PREVIOUS LINE
              3C A4     B5   0995  1914          TSTW    TTY$W_RB_TXTOFF(R4)           ; IS THERE ANY MORE DATA LEFT
                 E8     13   0998  1915          BEQL    57$                           ; NO THEN JUST MAKE SURE THE PROMPT
                        099A  1916                                                    ; ECHO'S
D8 48 A5         18     E1   099A  1917          BBC     #TT2$V_ANSICRT,UCB$L_DEVDEPND2(R5),55$; REECHO ON NON ANSI TERMINALS
              3A A4     B5   099F  1918          TSTW    TTY$W_RB_CPZORG(R4)           ; WAS THIS AT THE RIGHT
                 D3     12   09A2  1919          BNEQ    55$                           ; NO THEN USE THE OLD METHOD
3C A4         30 A4     A2   09A4  1920          SUBW    TTY$W_RB_LINOFF(R4),TTY$W_RB_TXTOFF(R4); DELETE THE LINE
```

```
                30 A4   B4  09A9  1921          CLRW    TTY$W_RB_LINOFF(R4)     ;
             64  2C A4   D1  09AC  1922          CMPL    TTY$L_RB_LIN(R4),TTY$L_RB_TXT(R4); DOES A PROMPT NEED TO BE ECHOED
                   0B   12  09B0  1923          BNEQ    60$                     ; NO THEN DON'T ECHO A PROMPT
             44 A4    07   9B  09B2  1924          MOVZBW  #TTY$K_ER_PRMECHO,TTY$W_RB_MODE(R4); SET UP TO ECHO THE PROMPT
                0B A4   94  09B6  1925          CLRB    TTY$B_RB_ECHLEN(R4)     ; NO STRING TO OUTPUT
                        09B9  1926          SET_STATE EDITREAD              ; TELL THE REST OF THE SYSTEM ABOUT THE PROM
                        09BD  1927
                38 A4   B4  09BD  1928 60$:     CLRW    TTY$W_RB_CPZCUR(R4)     ; CLEAR THE CURSOR POSITION
                        09C0  1929          SET_STATE SKIPCRLF              ; SKIP THE FREE LINEFEED
         00000000'EF    9E  09C4  1930          MOVAB   TTY$A_ANSI_UPCEL,R4     ; GET THE STRING ADDRESS
                    54      09CA
                069D    31  09CB  1931          BRW     STRTMULTI               ; AND DELETE THIS LINE
                        09CE  1932          .disable lsb
```

```
                     09CE  1934
                     09CE  1935 ;++
                     09CE  1936 ; DELCHAR - DELETE CHARACTER ROUTINE
                     09CE  1937 ;
                     09CE  1938 ; FUNCTIONAL DESCRIPTION:
                     09CE  1939 ;
                     09CE  1940 ; THIS ROUTINE DELETES THE LAST TYPED CHARACTER FROM THE READ BUFFER.
                     09CE  1941 ; THEN IT SETS UP THE PROPER ECHO SEQUENCE FOR THE DELETED CHARACTER.
                     09CE  1942 ;
                     09CE  1943 ; INPUTS:
                     09CE  1944 ;
                     09CE  1945 ;       R2 = ADDRESS OF TTY STATE VECTOR
                     09CE  1946 ;       R5 = UCB ADDRESS
                     09CE  1947 ;
                     09CE  1948 ; OUTPUTS:
                     09CE  1949 ;
                     09CE  1950 ;       R2 = ADDRESS OF TTY STATE VECTOR
                     09CE  1951 ;       R5 = UCB ADDRESS
                     09CE  1952 ;--
                     09CE  1953 DELCHAR:                                     ;
          54   78 A5  D0  09CE  1954         MOVL    UCB$L_SVAPTE(R5),R4     ; ADDRESS READ BUFFER BLOCK
                     09D2  1955 ;
                     09D2  1956 ; IF ESCAPE SEQUENCE IN PROGRESS THEN DELETE ENTIRE STRING
                     09D2  1957 ;
                     09D2  1958         IF_STATE ESC,25$                     ; DELETE ESCAPE SEQUENCE
        30 A4   B7   09D6  1959         DECW    TTY$W_RB_LINOFF(R4)          ; ANY DATA
           30   19   09D9  1960         BLSS    28$                          ; IF EQL THEN NO
     51    30 A4   3C  09DB  1961         MOVZWL  TTY$W_RB_LINOFF(R4),R1       ; GET THE POINTER
  53    2C B441   9A  09DF  1962         MOVZBL  @TTY$C_RB_LIN(R4)[R1],R3     ; GET THE CHARACTER
        53   0D   91  09E4  1963         CMPB    #TTY$C_CR,R3                 ; IS THIS CHARACTER A RETURN
           22   13  09E7  1964         BEQL    28$                          ; YES THEN ADJUST THE LINE
        3C A4   B7  09E9  1965         DECW    TTY$W_RB_TXTOFF(R4)          ; ADJUST POINTER
                     09EC  1966         IF_NOT_STATE NOECHO,5$               ; BRANCH IF ECHO
           00   E1  09F0  1967         BBC     #TT2$V_LOCALECHO,-
        6F 48 A5       09F2  1968                 UCB$L_DEVDEPND2(R5),30$ ; BRANCH IF NOT LOCAL ECHO
                     09F5  1969
                     09F5  1970 ;
                     09F5  1971 ; TEST FOR SPECIAL DELETES
                     09F5  1972 ;
        09   53   91  09F5  1973 5$:     CMPB    R3,#TTY$C_TAB                ; TAB?
             03   13  09F8  1974         BEQL    8$                           ; IF NEQ THEN OUTPUT JUST THE CHARACTER
          00A3   31  09FA  1975         BRW     55$
  65 44 A5    0C   E0  09FD  1976 8$:     BBS     #TT$V_SCOPE,UCB$L_DEVDEPEND(R5),40$; BR IF SCOPE
          00DB   31  0A02  1977         BRW     75$                          ; BR IF NOT SCOPE
                     0A05  1978
          0144   30  0A05  1979 25$:    BSBW    DELESCAPE                    ; DELETE ESCAPE SEQUENCE
          FB22   31  0A08  1980         BRW     MOVEREADATA                  ; AND GO BACK TO WORK
                     0A0B  1981
        30 A4   B4  0A0B  1982 28$:    CLRW    TTY$W_RB_LINOFF(R4)          ; MAKE THE COUNT ZERO
        32 A4   B5  0A0E  1983         TSTW    TTY$W_RB_LINREST(R4)         ; is there nothing left?
           51   12  0A11  1984         BNEQ    30$                          ; NO THEN JUST RETURN
        3C A4   B5  0A13  1985         TSTW    TTY$W_RB_TXTOFF(R4)          ; NO LINE WAS WRAPPED
           4C   13  0A16  1986         BEQL    30$                          ; IF THIS COUNT IS ZERO
        3C A4   B7  0A18  1987         DECW    TTY$W_RB_TXTOFF(R4)          ; REMOVE THE LAST CHARACTER
         05AE   30  0A1B  1988         BSBW    FIND_BOL                     ; YES THEN CALCULATE THE NEW BEGINNING
                     0A1E  1989                                              ; OF LINE
  33 48 A5    18   E1  0A1E  1990         BBC     #TT2$V_ANSICRT,UCB$L_DEVDEPND2(R5),120$; REECHO ON NON ANSI TERMINAL
```

```
                3A A4   B5   0A23  1991          TSTW    TTY$W_RB_CPZORG(R4)      ; WAS THIS AT THE RIGHT
                   2E   12   0A26  1992          BNEQ    120$                     ; NO THEN USE THE OLD METHOD
             44 A4  01   9B   0A28  1993          MOVZBW  #TTY$K_ER_CLRECHO,TTY$W_RB_MODE(R4); ECHO THE DATA
          0B A4  30 A4   90   0A2C  1994          MOVB    TTY$W_RB_CINOFF(R4),TTY$B_RB_ECHLEN(R4); AND THE STRING TO OUTPUT
          14 A4  2C A4   D0   0A31  1995          MOVL    TTY$L_RB_LIN(R4),TTY$L_RB_ECHSTR(R4); AND THE ADDRESS
             64  2C A4   D1   0A36  1996          CMPL    TTY$L_RB_LIN(R4),TTY$L_RB_TXT(R4); DOES A PROMPT NEED TO BE ECHOFD
                   04   12   0A3A  1997          BNEQ    110$                     ; NO THEN DON'T ECHO A PROMPT
             44 A4  07   9B   0A3C  1998          MOVZBW  #TTY$K_ER_PRMECHO,TTY$W_RB_MODE(R4); SET UP TO ECHO THE PROMPT
                         0A40  1999
                         0A40  2000 110$:   SET_STATE <SKIPCRLF,EDITREAD>  ; SKIP THE FREE LINEFEED
             38 A4  02   AE   0A48  2001          MNEGW   #2,TTY$W_RB_CPZCUR(R4)  ; DON'T RESET CURSOR
          00000000'EF    9E   0A4C  2002          MOVAB   TTY$A_ANSI_OPCEL,R4     ; GET THE STRING ADDRESS
                   54          0A52
                 0615   31   0A53  2003          BRW     STRTMULTI                ; AND DELETE THIS LINE
                         0A56  2004
                         0A56  2005 120$:   SET_STATE <EDITREAD>             ; RESTORE THE LINE TO IT'S FULL VIGOR
          00000000'EF    9E   0A5A  2006          MOVAB   TTY$A_CTRLR,R4           ; GET A NEW LINE
                   54          0A60
                 0607   31   0A61  2007          BRW     STRTMULTI                ; and return to the main loop
                         0A64  2008
                 FAC6   31   0A64  2009 30$:    BRW     MOVEREADATA              ; VECTOR TO CONTINUE
                         0A67  2010
                         0A67  2011 ;
                         0A67  2012 ; DELETE A TAB ON A SCOPE
                         0A67  2013 ;
             00FC C5    B5   0A67  2014 40$:    TSTW    UCB$W_TT_CURSOR(R5)      ; Have we wrapped?
                   47   13   0A6B  2015          BEQL    60$                      ; Branch to simulate CTRL-R, if yes.
                 048A   30   0A6D  2016          BSBW    BACK_TAB                 ; GET THE TAB LENGTH
             00FC C5    B1   0A70  2017          CMPW    UCB$W_TT_CURSOR(R5), UCB$W_DEVBUFSIZ(R5) ; At extreme right?
                42 A5          0A74
                   11   1F   0A76  2018          BLSSU   45$                      ; Branch if within screen bounds.
                   1C   1A   0A78  2019          BGTRU   47$                      ; Branch if beyond right edge.
                   53   D6   0A7A  2020          INCL    R3                       ; Adjust number of backspaces for
              FFFFFFF8  8F   CA   0A7C  2021          BICL    #^C7, R3                 ; right-hand edge effects.
                   53          0A82
                   41   13   0A83  2022          BEQL    66$                      ; If no backspace, just delete at edge.
                00FC  C5   B7   0A85  2023          DECW    UCB$W_TT_CURSOR(R5)      ; Adjust cursor value for one less BS.
                07    53   A3   0A89  2024 45$:    SUBW3   R3,#7,UCB$W_TT_MULTILEN(R5); GET THE LENGTH CORRECTLY
                00DC  C5          0A8C
          54    0001'CF    9E   0A8F  2025          MOVAB   W^TTY$A_DELCRTTAB+1,R4; Get address backspaces string.
                   45   11   0A94  2026          BRB     70$                      ; Go output them.
                         0A96  2027
                53    07   A2   0A96  2028 47$:    SUBW    #7, R3                   ; Adjust off-the-screen cursor
             00FC C5    53   A0   0A99  2029          ADDW    R3, UCB$W_TT_CURSOR(R5) ; position for deleted tab.
                   26   11   0A9E  2030          BRB     66$                      ; Go do delete-at-edge.
                         0AA0  2031
                         0AA0  2032 ;
                         0AA0  2033 ; NORMAL CHARACTER DELETE
                         0AA0  2034 ;
                         0AA0  2035 55$:
                   F0  8F   93   0AA0  2036          BITB    #<TTY$M_CH_CTRL!TTY$M_CH_SPEC!TTY$M_CH_CTRL2!TTY$M_CH_CTRL3>,-
             0000'CF43          0AA3  2037          W^TTY$A_TYPE[R3]                 ; TEST FOR SPECIAL
                   14   12   0AA7  2038          BNEQ    62$                      ; NON PRINTING CHARACTER
                         0AA9  2039
          32 44 A5   0C   E1   0AA9  2040          BBC     #TT$V_SCOPE,UCB$L_DEVDEPEND(R5),75$; BR IF NOT SCOPE
                         0AAE  2041 ;
                         0AAE  2042 ; IF THE CURSOR IS AT THE LEFT MARGIN AND DATA IS PRESENT,
```

```
                        0AAE  2043 ; FORCE A CONTROL R IN STEAD OF A BACKSPACE.
                        0AAE  2044 ;
            00FC C5  B5 0AAE  2045 59$:    TSTW    UCB$W_TT_CURSOR(R5)      ; CURSOR AT LEFT MARGIN?
                 12  12 0AB2  2046        BNEQ    66$                      ; BR IF NORMAL BACKSPACE NEEDED
                        0AB4  2047 ;
                        0AB4  2048 ; FORCE CONTROL R FOR RUBOUT RESPONSE
                        0AB4  2049 ;
                        0AB4  2050 60$:
        54    78 A5  D0 0AB4  2051        MOVL    UCB$L_SVAPTE(R5),R4      ; GET READ PACKET ADDRESS
              3C A4  B5 0AB8  2052        TSTW    TTY$W_RB_TXTOFF(R4)      ; DATA PRESENT?
                 03  12 0ABB  2053        BNEQ    63$                      ; IF DATA PRESENT, ECHO IT IN NORMAL FASSION
               FA6D  31 0ABD  2054 62$:    BRW     MOVEREADATA              ; IF NO DATA THEN NO RESPONSE
               FF93  31 0AC0  2055 63$:    BRW     120$                     ; DATA TO ECHO THEN ECHO
               0290  31 0AC3  2056 65$:    BRW     EDITREAD                 ; START THE OUTPUT
                        0AC6  2057 ;
                        0AC6  2058 ; NORMAL BACKSPACE RUBOUT RESPONSE
                        0AC6  2059 ;
            53    20  9A 0AC6  2060 66$:    MOVZBL  #TTY$C_BLANK,R3          ; ASSUME SPACE IS FIRST CHARACTER
            00FC C5  B1 0AC9  2061        CMPW    UCB$W_TT_CURSOR(R5),UCB$W_DEVBUFSIZ(R5); AND EXTREME RIGHT?
              42 A5
                 74  1E 0ACD  2062        BGEQU   100$                      ; IF YES THEN ONLY OUTPUT SPACE
                        0AD1  2063 ;
                        0AD1  2064 ; DELETE IS IN MID SCREEN
                        0AD1  2065 ;
        54  0000'CF  9E 0AD1  2066        MOVAB   W^TTY$A_SPACEBACK,R4     ; ADDRESS BACKSPACE STRING
            00DC C5  84 9B 0AD6  2067        MOVZBW  (R4)+,UCB$W_TT_MULTILEN(R5);AND THEN IT'S LENGTH
            53    08  9A 0ADB  2068 70$:    MOVZBL  #TTY$C_BS,R3             ; START STRING WITH BACKSPACE
                 15  11 0ADE  2069        BRB     90$                      ; START OUTPUT
        5D 04 A2  01  E2 0AE0  2070 75$:    BBSS    #TTY$V_ST_DEL,4(R2),95$  ; IF NOT FIRST TIME THEN OUTPUT CHARACTER
              54  18 A5  9E 0AE5  2071        MOVAB   UCB$W_BUFQUO(R5),R4      ; ADDRESS STRING SPACE
            00DC C5  01  B0 0AE9  2072        MOVW    #1,UCB$W_TT_MULTILEN(R5)         ; SETUP THE LENGTH OF THE MULTIECHO
            64    53  9B 0AEE  2073        MOVZBW  R3,(R4)                  ; AND PUT THE CHARACTERS THERE
                        0AF1  2074 ;
                        0AF1  2075 ; OUTPUT "\" TO START OR END DELETE SEQUENCE
                        0AF1  2076 ;
          53    5C  8F  9A 0AF1  2077 80$:    MOVZBL  #^A@\@,R3                ; FOR FIRST TIME OUTPUT "\"
        00D8 C5    54  D0 0AF5  2078 90$:    MOVL    R4,UCB$L_TT_MULTI(R5)    ; ADDRESS STRING
                        0AFA  2079        SET_STATE MULTI                  ;
          54    78 A5  D0 0AFE  2080        MOVL    UCB$L_SVAPTE(R5),R4      ; RESTORE THE READ PACKET ADDRESS
                32 A4  B5 0B02  2081        TSTW    TTY$W_RB_LINREST(R4)     ; ANY CHARACTERS TO THE RIGHT
                   3B  13 0B05  2082        BEQL    95$                      ; NO THEN HANDLE NORMALY
                        0B07  2083
                   3F  BB 0B07  2084        PUSHR   #^M<R0,R1,R2,R3,R4,R5>   ; SAVE NECESSARY STATE
                        0B09  2085        IF_STATE NOECHO,92$
                        0B0D  2086        SET_STATE <EDITREAD>             ; SET THE EDIT ECHOING STATE
          51    30 A4  3C 0B11  2087 92$:    MOVZWL  TTY$W_RB_LINOFF(R4),R1   ; GET THE OFFSET TO THIS CHARACTER
              2C B441  9E 0B15  2088        MOVAB   @TTY$L_RB_LIN(R4)[R1],TTY$L_RB_ECHSTR(R4); SETUP THE ADDRESS TO STAR
                14 A4       0B19
          53    14 B4  9A 0B1B  2089        MOVZBL  @TTY$L_RB_ECHSTR(R4),R3 ; GET THE CHARACTER BACK
          44 A4    01  98 0B1F  2090        MOVZBW  #TTY$K_ER_CLRECHO,TTY$W_RB_MODE(R4); CLEAR TO THE END OF LINE THEN E
        0B A4    32 A4  90 0B23  2091        MOVB    TTY$W_RB_LINREST(R4),TTY$B_RB_ECHLEN(R4); AND THE LENGTH
                51  01  CE 0B28  2092        MNEGL   #1,R1                    ; setup to find backward size
                  042F  30 0B2B  2093        BSBW    CHAR_SIZE                ; FIND THE SIZE OF THIS CHARACTER
            00FC C5  53  A3 0B2E  2094        SUBW3   R3,UCB$W_TT_CURSOR(R5),TTY$W_RB_CPZCUR(R4)       ; SUBTRACT OUT THE L
                38 A4       0B33
                52  01  9A 0B35  2095        MOVZBL  #1,R2                    ; SETUP R2 AS 1 CHARACTER
                32 A4  28 0B38  2096        MOVC3   TTY$W_RB_LINREST(R4),-
```

```
                14 B442      0833  2097                    @TTY$L_RB_ECHSTR(R4)[R2],@TTY$L_RB_ECHSTR(R4); COMPRESS THE STRING
                  14 B4      083E
                     3F  BA  0840  2098          POPR    #^M<R0,R1,R2,R3,R4,R5>  ; SAVE NECESSARY STATE
                                 0842  2099
                   F649  31  0842  2100 95$:    BRW     FORMAT_CHAR             ; START UP OUTPUT
                                 0845  2101 100$:
                00FC C5  B7  0845  2102          DECW    UCB$W_TT_CURSOR(R5)     ; ADJUST CURSOR
                   F603  31  0849  2103          BRW     OUTPUTANDWAIT           ; VECTOR FOR SPACE ONLY RESPONSE
```

```
                        0B4C  2105
                        0B4C  2106 ;++
                        0B4C  2107 ; DELETE AN ESCAPE SEQUENCE IN PROGRESS
                        0B4C  2108 ;
                        0B4C  2109 ; THIS ROUTINE IS USED BY THE CONTROL U AND DELETE LOGIC TO RESET
                        0B4C  2110 ; AND DELETE THE ESCAPE SEQUENCE IN PROGRESS
                        0B4C  2111 ;
                        0B4C  2112 DELESCAPE:                                   ; DELETE CURRENT ESCAPE SEQUENCE
   51    58 A5    D0    0B4C  2113          MOVL    UCB$L_IRP(R5),R1           ; GET ADDRESS OF PACKET.
 3C A4   3A A1    A2    0B50  2114          SUBW    IRP$L_MEDIA+2(R1),TTY$W_RB_TXTOFF(R4); ADJUST TRANSFER SIZE FOR SEQU
         38 A1    D4    0B55  2115          CLRL    IRP$L_MEDIA(R1)            ; ESCAPE LENGTH
                        0B58  2116          CLR_STATE <ESC,BADESC>             ; CLEAR ESCAPE
                  05    0B5E  2117 10$:     RSB                                ; RETURN
```

```
                        0B5F  2119 ;++
                        0B5F  2120 ; DELETE_WORD
                        0B5F  2121 ;
                        0B5F  2122 ; DESCRIPTION:
                        0B5F  2123 ;       DELETE WORD TO THE LEFT OF THE CURSOR
                        0B5F  2124 ;--
             FE6C   31  0B5F  2125 GODEL:  BRW     DELCHAR                    ; ACT AS IF A DELETE WAS TYPED
             F73B   31  0B62  2126 DROP_CHAR:BRW   DROP                       ; DROP THE CHARACTER
                        0365  2127 DELETE_WORD:
       53    30 A4   3C  0B65  2128         MOVZWL  TTY$W_RB_LINOFF(R4),R3    ; GET THE OFFSET
                53   D7  0B69  2129         DECL    R3                        ; SUBTRACT 1
                F5   19  0B6B  2130         BLSS    DROP_CHAR                 ; no characters then handle suchly
     53    2C B443   9A  0B6D  2131         MOVZBL  @TTY$L_RB_LIN(R4)[R3],R3; GET THE CHARACTER AT THIS POINT
                53   E0  0B72  2132         BBS     R3,TTY$A_PREFIX,GODEL; IF THIS CHARACTER
  E5 00000000'EF          0B74
                        0B7A  2133                                           ; IS A PREFIX THEN DELETE ONLY IT
                37   BB  0B7A  2134         PUSHR   #^M<R0,R1,R2,R4,R5>
       50    30 A4   3C  0B7C  2135         MOVZWL  TTY$W_RB_LINOFF(R4),R0    ; GET THE CURRENT POSITION
                50   DD  0B80  2136         PUSHL   R0                        ; SAVE THE OFFSET
           51   01   CE  0B82  2137         MNEGL   #1,R1                     ; WE ARE MOVING BACKWARDS
                52   D4  0B85  2138         CLRL    R2                        ; CLEAN OUT THE CHARACTER POSITION COUNTER
                50   D7  0B87  2139 10$:    DECL    R0                        ; MOVE ONE CHARACTER TO THE LEFT
                34   19  0B89  2140         BLSS    30$                       ; WHEN DONE SAY SO
     53    2C B440   9A  0B8B  2141         MOVZBL  @TTY$L_RB_LIN(R4)[R0],R3; GET A CHARACTER
                53   E1  0B90  2142         BBC     R3,TTY$A_WORDTERM,20$     ; SKIP OVER TERMINATORS
  0C 00000000'EF          0B92
       30 A4    50   B0  0B98  2143         MOVW    R0,TTY$W_RB_LINOFF(R4)    ; UPDATE LINOFF TO MAKE TAB DELETION WORK
             03BE   30  0B9C  2144         BSBW    CHAR_SIZE                 ; GET THE LENGTH OF THIS CHARACTER
           52   53   C0  0B9F  2145         ADDL    R3,R2                     ; UPDATE CURSOR POSITION
                E3   11  0BA2  2146         BRB     10$                       ; AND CONTINUE
                        0BA4  2147
       30 A4    50   B0  0BA4  2148 20$:    MOVW    R0,TTY$W_RB_LINOFF(R4)    ; UPDATE LINOFF TO MAKE TAB DELETION WORK
             03B2   30  0BA8  2149         BSBW    CHAR_SIZE                 ; GET THE LENGTH OF THIS CHARACTER
           52   53   C0  0BAB  2150         ADDL    R3,R2                     ; UPDATE THE POSITION
                50   D7  0BAE  2151         DECL    R0                        ; KEEP MOVING
                0D   19  0BB0  2152         BLSS    30$                       ; UNTILL THE END
     53    2C B440   9A  0BB2  2153         MOVZBL  @TTY$L_RB_LIN(R4)[R0],R3; GET ANOTHER CHARCTER
                53   E1  0BB7  2154         BBC     R3,TTY$A_WORDTERM,20$     ; KEEP MOVING UNTILL NOT A TERMINATOR
  E5 00000000'EF          0BB9
                        0BBF  2155
                50   D6  0BBF  2156 30$:    INCL    R0                        ; GO FOWARD A CHARACTER
     0100 C5    52   B0  0BC1  2157         MOVW    R2,UCB$W_TT_BSPLEN(R5)    ; ECHO THE RIGHT NUMBER OF BACKSPACES
     00FC C5    ??   A3  0BC6  2158         SUBW3   R2,UCB$W_TT_CURSOR(R5),TTY$W_RB_CPZCUR(R4); SETUP THE CURSOR POSITIO
           38 A4          0BCB
                51 8ED0  0BCD  2159         POPL    R1                        ; GET THE CURRENT POSITION
     52    51   50   A3  0BD0  2160         SUBW3   R0,R1,R2                  ; GET THE NUMBER OF CHARACTERS DELETED
        3C A4   52   A2  0BD4  2161         SUBW    R2,TTY$W_RB_TXTOFF(R4)    ; AND REMOVE THESE CHARACTERS FROM THE COUNT
           2C B440   9E  0BD8  2162         MOVAB   @TTY$L_RB_LIN(R4)[R0],TTY$L_RB_ECHSTR(R4); GET THE ADDRESS FOR UPDAT
             14 A4       0BDC
             32 A4   28  0BDE  2163         MOVC3   TTY$W_RB_LINREST(R4),-    ; DELETE THE WORD
           2C B441       0BE1  2164                 @TTY$L_RB_LIN(R4)[R1],@TTY$L_RB_ECHSTR(R4);
             14 B4       0BE4
                37   BA  0BE6  2165         POPR    #^M<R0,R1,R2,R4,R5>       ; RESTORE THE STATE
                        0BE8  2166         IF_STATE NOECHO,40$
                        0BEC  2167         SET_STATE <BACKSPACE,EDITREAD>    ; BACKSPACE OVER THE WORD THEN ECHO EVERYTHI
        44 A4   01   9B  0BF1  2168         MOVZBW  #TTY$K_ER_CLRECHO,TTY$W_RB_MODE(R4); SETUP THE MODE FOR EDITECHOING
  0B A4    32 A4   90  0BF5  2169         MOVB    TTY$W_RB_LINREST(R4),TTY$B_RB_ECHLEN(R4); AND THE LENGTH TO ECHO
```

```
F408    31  0BFA  2170 40$:    BRW     GETNEXTCHAR             ; GET THE NEXT CHARACTER
            0BFD  2171
```

```
                        0BFD  2173 ;++
                        0BFD  2174 ; Escape_Char
                        0BFD  2175 ;
                        0BFD  2176 ;       Possible escape character found, could be an altmode for
                        0BFD  2177 ; lower case terminals or could be a CSI for new eight bit terminals.
                        0BFD  2178 ; All cases must be handled apropriately.
                        0BFD  2179 ;--
                        0BFD  2180 ESCAPE_CHAR:
     53    7D 8F    91   0BFD  2181         CMPB    #TTY$C_LOWESC1,R3       ; ALT MODE?
           06       13   0C01  2182         BEQL    47$                     ; YES
     53    7E 8F    91   0C03  2183         CMPB    #TTY$C_LOWESC2,R3       ; ALT MODE?
           08       12   0C07  2184         BNEQ    52$                     ; YES
39 44 A5   07       E0   0C09  2185 47$:    BBS     #TT$V_LOWER,UCB$L_DEVDEPEND(R5),51$; LOWER CASE TERMINAL CONTINUE
        53 1B       9A   0C0E  2186         MOVZBL  #TTY$C_ESCAPE,R3        ; CHANGE CHARACTER TO ESCAPE
                        0C11  2187 52$:
                        0C11  2188 ;
                        0C11  2189 ; PROCESS ESCAPE SEQUENCES AND ESCAPES
                        0C11  2190 ;
                        0C11  2191         IF_NOT_STATE <ESCAPE,EDITING>,55$; NOT AN ESCAPE TERMINAL THEN EXIT
           F887     30   0C19  2192         BSBW    ESCINIT                 ; HANDLE THE ESCAPE PREFIXES
0103 C5    51       90   0C1C  2193         MOVB    R1,UCB$B_TT_ESC(R5)
        48 A4       B4   0C21  2194         CLRW    TTY$W_RB_ESCTKN(R4)
     51 58 A5       D0   0C24  2195         MOVL    UCB$L_IRP(R5),R1        ; GET THE IRP ADDRESS
     38 A1  53      B0   0C28  2196         MOVW    R3,IRP$L_MEDIA(R1)      ; SET TERMINATOR CHARACTER
        3A A1       B6   0C2C  2197         INCW    IRP$L_MEDIA+2(R1)       ; START TERMINATOR SIZE
     51 3C A4       3C   0C2F  2198         MOVZWL  TTY$W_RB_TXTOFF(R4),R1  ; GET THE PLACE TO STORE THE ESCAPE
00 6441    1B       90   0C33  2199         MOVB    #TTY$C_ESCAPE,@TTY$L_RB_TXT(R4)[R1]; AND PLACE THE CHARACTER IN THE
        3C A4       B6   0C38  2200         INCW    TTY$W_RB_TXTOFF(R4)     ; AND INCREMENT THE WORLD
                        0C3B  2201         IF_STATE ESC,60$
                        0C3F  2202         SET_STATE ESC                   ; BEGIN ESCAPE SEQUENCE
           F953     31   0C44  2203         BRW     CHECK_BUFFER            ; INSERT THE ESCAPE AT THE END OF THE BUFFER
                        0C47  2204
                        0C47  2205 51$:
                        0C47  2206         IF_NOT_STATE ESC,55$
     FA18  31           0C4B  2207         BRW     ESCINPROG               ; ESCAPE IN PROGGRSS THEN CONTINUE
                        0C4E  2208 55$:
     F927  31           0C4E  2209         BRW     INSERT_CHAR             ; ELSE INSERT THE CHARACTER
                        0C51  2210 ;
                        0C51  2211 ; ALREADY IN ESCAPE MODE THEN MAKE THIS BAD
                        0C51  2212 ;
                        0C51  2213 60$:    SET_STATE <BADESC,EOL>          ;
     F3A9  31           0C59  2214         BRW     GETNEXTCHAR
```

```
                     0C5C  2216
                     0C5C  2217 ;++
                     0C5C  2218 ; FORWARD_CHAR
                     0C5C  2219 ;       MOVE THE CURSOR FORWARD 1 CHARACTER POSITION.
                     0C5C  2220 ;--
                     0C5C  2221 FORWARD_CHAR:
  51    30 A4    3C  0C5C  2222          MOVZWL  TTY$W_RB_LINOFF(R4),R1  ; GET THE OFFSET TO THE CHARACTER
53    2C B441    9A  0C60  2223          MOVZBL  @TTY$[_RB_LIN(R4)[R1],R3; GET THE CHARCTER
      32 A4      B7  0C65  2224          DECW    TTY$W_RB_[INREST(R4)    ; FORWARD A CHARACTER
         27      19  0C68  2225          BLSS    50$                     ; AT THE END THEN DON'T ALLOW IT
         51      D4  0C6A  2226          CLRL    R1                      ; GOING FORWARD
       02EE      30  0C6C  2227          BSBW    CHAR_SIZE               ; CHECK THE SIZE OF THE CHARACTER
00FC C5    53    A1  0C6F  2228          ADDW3   R3,UCB$W_TT_CURSOR(R5),TTY$W_RB_CP?CUR(R4); UPDATE THE CURSOR POSITI
      38 A4        0C74
  51    30 A4    3C  0C76  2229          MOVZWL  TTY$W_RB_LINOFF(R4),R1  ; GET THE OFFSET TO THE CHARACTER
53    2C B441    9A  0C7A  2230          MOVZBL  @TTY$[_RB_LIN(R4)[R1],R3; GET THE CHARCTER
      30 A4      B6  0C7F  2231          INCW    TTY$W_RB_[INOFF(R4)     ; UPDATE THE REST POINTER
      53   09    91  0C82  2232          CMPB    #TTY$C_TAB,R3           ; ARE WE FORWARDING OVER A TAB?
         03      12  0C85  2233          BNEQ    10$                     ; NO THEN CONTINUE
       040E      30  0C87  2234          BSBW    TABRIGHT                ; YES THEN CHECK FOR A TAB TO THE RIGHT
                     0C8A  2235 10$:     IF_STATE NOECHO,GODROP
       F4FD      31  0C8E  2236          BRW     FORMAT_CHAR             ; THEN OUPUT THE CHARACTER TO GO FOWARD
                     0C91  2237
      32 A4      B4  0C91  2238 50$:     CLRW    TTY$W_RB_LINREST(R4)    ; RESET THE POINTER
       F609      31  0C94  2239 GODROP:  BRW     DROP                    ; AND DROP THE CHARACTER
```

```
                        0C97  2241 ;++
                        0C97  2242 ; MOVE_BOL
                        0C97  2243 ;
                        0C97  2244 ; DESCRIPTION:
                        0C97  2245 ;       GO TO THE BEGINNING OF THE LINE.
                        0C97  2246 ;--
                        0C97  2247 MOVE_BOL:
          3A A4   B5    0C97  2248          TSTW    TTY$W_RB_CPZORG(R4)     ; START AT ZERO?
             03   13    0C9A  2249          BEQL    10$                     ; YES THEN NO SPECAIL ACTION
           0484   30    0C9C  2250          BSBW    ZERO_CPZORG             ; ZERO ORG CURSOR IF WE HAVE WRAPPED
32 A4      30 A4   A0    0C9F  2251 10$:     ADDW    TTY$W_RB_LINOFF(R4),TTY$W_RB_LINREST(R4); MOVE THE CHARACTER COUNTS
          30 A4   B4    0CA4  2252          CLRW    TTY$W_RB_LINOFF(R4)     ; AND CLEAR THE NUMBER OF CHARACTERS WE HAVE
           03EE   30    0CA7  2253          BSBW    TABRIGHT                ; CHECK FOR TABS TO THE RIGHT
   64      2C A4   D1    0CAA  2254          CMPL    TTY$L_RB_LIN(R4),TTY$L_RB_TXT(R4); ARE WE ON THE FIRST LINE
             0C   12    0CAE  2255          BNEQ    20$                     ; NO  THEN JUST DROP IT
   0B A4     01   8E    0CB0  2256          MNEGB   #1,TTY$B_RB_ECHLEN(R4)  ; CLEAN OUT THE STRING TO ECHO
   44 A4     07   9B    0CB4  2257          MOVZBW  #TTY$K_ER_PRMECHO,TTY$W_RB_MODE(R4); SETUP THE MODE
                        0CB8  2258          SET_STATE <EDITREAD>
          3A A4   B5    0CBC  2259 20$:     TSTW    TTY$W_RB_CPZORG(R4)     ; START AT ZERO?
             1D   12    0CBF  2260          BNEQ    50$                     ; YES THEN NO SPECAIL ACTION
 00000001'EF      9E    0CC1  2261          MOVAB   TTY$A_ANSI_DEOL+1,UCB$L_TT_MULTI(R5); GET THE ADDRESS OF A <CR>
       00D8 C5        0CC7
 00DC C5   01   D0    0CCA  2262          MOVL    #1,UCB$W_TT_MULTILEN(R5); AND ONLY OUTPUT THE <CR>
                        0CCF  2263          SET_STATE <MULTI,SKIPLF,SKIPCRLF>; SETUP TO DO A MULTIECHO
          38 A4   B4    0CD9  2264          CLRW    TTY$W_RB_CPZCUR(R4)     ; MOVE TO THE BEGINNING WHEN DONE
             B6   11    0CDC  2265          BRB     GODROP                  ; THEN DISPATCH
                        0CDE  2266 ;
                        0CDE  2267 ; IF THE ORIGION IS NON-ZERO THEN MOVE THE CURSOR TO THE ORIGIONAL POSITION
                        0CDE  2268 ;
38 A4     3A A4   B0    0CDE  2269 50$:     MOVW    TTY$W_RB_CPZORG(R4),TTY$W_RB_CPZCUR(R4); SETUP TO MOVE BACK
           03CE   31    0CE3  2270          BRW     UPDATE_CURSOR
```

```
                        0CE6  2272 ;++
                        0CE6  2273 ; MOVE_EOL
                        0CE6  2274 ;
                        0CE6  2275 ; DESCRIPTION:
                        0CE6  2276 ;       MOVE TO THE END OF THE LINE.
                        0CE6  2277 ;--
                        0CE6  2278 MOVE_EOL:
0B A4    32 A4    90    0CE6  2279         MOVB    TTY$W_RB_LINREST(R4),TTY$B_RB_ECHLEN(R4); GET THE LENGTH TO ECHO
   51    30 A4    3C    0CEB  2280         MOVZWL  TTY$W_RB_LINOFF(R4),R1  ; GET THE OFFSET OF THIS CHARACTER
      2C B441    9E    0CEF  2281         MOVAB   @TTY$L_RB_LIN(R4)[R1],TTY$L_RB_ECHSTR(R4); SETUP THE ADDRESS ALSO
         14 A4         0CF3
   44 A4    02    9B    0CF5  2282         MOVZBW  #TTY$K_ER_ECHLINE,TTY$W_RB_MODE(R4); AND SETUP THE MODE
                        0CF9  2283         CLR_STATE TABRIGHT                  ; NO MORE CHARACTERS TO THE RIGHT SO
                        0CFE  2284                                             ; NO TABS EITHER
                        0CFE  2285         IF_STATE NOECHO,10$                 ; NO ECHO THEN DON'T ECHO
                        0D02  2286         SET_STATE EDITREAD                  ; THEN THE STATE
30 A4    32 A4    A0    0D06  2287 10$:    ADDW    TTY$W_RB_LINREST(R4),TTY$W_RB_LINOFF(R4); FIX THE COUNTS
         32 A4    B4    0D0B  2288         CLRW    TTY$W_RB_LINREST(R4)        ;...
          F2F4    31    0D0E  2289         BRW     GETNEXTCHAR                 ; GET THE NEXT CHARACTER
```

```
                     0D11  2291 ;++
                     0D11  2292 ; QUOTING
                     0D11  2293 ;
                     0D11  2294 ; DESCRIPTION:
                     0D11  2295 ;       INITIATE QUOTING SEQUENCE
                     0D11  2296 ;--
                     0D11  2297 QUOTING:
                     0D11  2298         SET_STATE QUOTING               ;SET THE STATE QUOTE
F2EC   31            0D16  2299         BRW     GETNEXTCHAR             ; AND CONTINUE ON
```

```
                          0D19  2301 ;++
                          0D19  2302 ; RECALL
                          0D19  2303 ;
                          0D19  2304 ; DESCRIPTION:
                          0D19  2305 ;       SETUP THE NECESSARY STATE TO RECALL THE LAST COMMAND.
                          0D19  2306 ;--
                          0D19  2307 RECALL:
28 20 A4    10    E0      0D19  2308         BBS     #TRM$V_TM_NORECALL,TTY$L_RB_MOD(R4),20$
 54    00E4 C5    D0      0D1E  2309         MOVL    UCB$L_TT_TYPAHD(R5),R4  ;GET THE ADDRESS OF THE TYPEAHEA BUFFER
       14 A4      B5      0D23  2310         TSTW    TTY$W_TA_RCLSIZ(R4)     ; CHECK THE SIZE OF THE BUFFER
          1B      13      0D26  2311         BEQL    10$                     ; DON'T DO ANYTHING IF NO CHARACTERS
       0E A4      B4      0D28  2312         CLRW    TTY$W_TA_RCLOFF(R4)     ;CLEAN OUT THE OFFSET TO THE STRING
                          0D2B  2313         SET_STATE <RECALL,EDITREAD>     ;TELL THE DISPATCHER THAT WE ARE RECALLING
    54    78 A5    D0      0D2F  2314         MOVL    UCB$L_SVAPTE(R5),R4     ; GET THE PACKET ADDRESS
       3C A4      B4      0D33  2315         CLRW    TTY$W_RB_TXTOFF(R4)     ; NO MORE TEXT
       30 A4      B4      0D36  2316         CLRW    TTY$W_RB_LINOFF(R4)     ; CLEAR THE LINE
       32 A4      B4      0D39  2317         CLRW    TTY$W_RB_LINREST(R4)    ; EVERYTHING ELSE
 2C A4    64      D0      0D3C  2318         MOVL    TTY$L_RB_TXT(R4),TTY$L_RB_LIN(R4)
          FB90    31      0D40  2319         BRW     CTRLR                   ; THEN CONTROL-R THE READ
          F2BF    31      0D43  2320 10$:    BRW     GETNEXTCHAR
          F82F    31      0D46  2321 20$:    BRW     INSERT_CHAR
```

```
                     0D49  2323 ;++
                     0D49  2324 ; TOGGELINSOV
                     0D49  2325 ;
                     0D49  2326 ; Description
                     0D49  2327 ;       Toggel the insert/overstrike mode flag
                     0D49  2328 ;--
                     0D49  2329 TOGGELINSOV:
05 04 A2    17   E4 0D49  2330         BBSC    #TTY$V_ST_OVERSTRIKE,4(R2),10$; TOGGEL THE OVERSTRIKE BIT
                     0D4E  2331         SET_STATE OVERSTRIKE
          F2AF   31 0D53  2332 10$:    BRW     GETNEXTCHAR
```

```
                     0D56  2335          .SBTTL  EDITREAD - READ EDITING STATE
                     0D56  2336
                     0D56  2337 ;++
                     0D56  2338 ; EDITREAD - STATE TO ALLOW A READ EDITING SEQUENCE
                     0D56  2339 ;
                     0D56  2340 ; FUNCTIONAL DESCRIPTION:
                     0D56  2341 ;
                     0D56  2342 ;       WHEN ECHOING PARTS OF READS IT IS NECESSARY TO ECHO SEVERAL
                     0D56  2343 ; STRINGS, THIS ROUTINE FILLS THAT NEED
                     0D56  2344 ;
                     0D56  2345 ; INPUTS:
                     0D56  2346 ;
                     0D56  2347 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                     0D56  2348 ;       R5 = UCB ADDRESS
                     0D56  2349 ;
                     0D56  2350 ; OUTPUTS:
                     0D56  2351 ;
                     0D56  2352 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                     0D56  2353 ;       R4 = ADDRESS OF THE READ PACKET
                     0D56  2354 ;       R5 = UCB ADDRESS
                     0D56  2355 ;
                     0D56  2356 ;--
                     0D56  2357 EDITREAD:                                    ; READ EDITING
          54   78 A5  D0  0D56  2358          MOVL    UCB$L_SVAPTE(R5),R4    ; GET THE READ PACKET ADDRESS
                     0D5A  2359          CASE    TTY$W_RB_MODE(R4),TYPE=W,<-
                     0D5A  2360                  NORMAL,-
                     0D5A  2361                  CLRECHO,-                  ; ECHO WITH CLEAR BIT SET
                     0D5A  2362                  ECHLINE,-                  ; JUST ECHO WHAT IS SPECIFYED
                     0D5A  2363                  UPDCURSOR,-                ; UPDATE THE CURSOR POSITION
                     0D5A  2364                  EXITING,-                  ; SETUP TO EXIT.
                     0D5A  2365                  MOVECURSOR,-               ; ALLOW CURSOR TO BE UPDATED
                     0D5A  2366                  CLRREST,-                  ; CLEAR THE REST OF THE LINE
                     0D5A  2367                  PRMECHO,-                  ; ECHO PROMPT THEN CONTINUE
                     0D5A  2368                  PRMECHO1,-                 ; CONTINUING STATE FOR PROMPTING
                     0D5A  2369                  AESECHO,-                  ; ECHO AES STRING AND NOSTRING
                     0D5A  2370                  RVECHO,-                   ; echo in the readverify way
                     0D5A  2371                  SIMCEOL>                   ; SIMULATE CLEAR TO END OF LINE
                     0D77  2372 NORMAL:
                     0D77  2373
                     0D77  2374          IF_STATE <CTRLR,RDVERIFY>,120$          ; HAVE WE COMPLETED THE CONTROL-R
        3A A4   B5   0D83  2375          TSTW    TTY$W_RB_CPZORG(R4)        ; START AT ZERO?
            10  13   0D86  2376          BEQL    25$                        ; YES THEN NO SPECAIL ACTION
     00FC C5   B5   0D88  2377          TSTW    UCB$W_TT_CURSOR(R5)        ; AND WE ARE NOW AT ZERO
            0A  12   0D8C  2378          BNEQ    25$                        ; YES THEN WE SHOULD FIXUP THE LINE
        3A A4   B4   0D8E  2379          CLRW    TTY$W_RB_CPZORG(R4)        ; DO THE FIXING
          0238  30   0D91  2380          BSBW    FIND_BOL                   ; FIND THE BEGINNING OF THE LAST LINE
     00FC C5   B4   0D94  2381          CLRW    UCB$W_TT_CURSOR(R5)        ; FIX THE CURSOR (CHANGED DURING FIND_BOL)
                     0D98  2382 25$:     IF_NOT_STATE SKIPCRLF,119$         ; NO THEN ARE WE DOING A CR <CEL>
                     0D9C  2383          SET_STATE <SKIPLF>                 ; YES THEN SKIP A LEADING LINEFEED AND CONTI
     44 A4    06   9B   0DA0  2384          MOVZBW  #TTY$K_ER_CLRREST,TTY$W_RB_MODE(R4); THEN CLEAR THE REST
        32 A4   B5   0DA4  2385          TSTW    TTY$W_RB_CINREST(R4)       ; CURSOR VALID?
            04  12   0DA7  2386          BNEQ    119$                       ; YES THEN USE IT
     38 A4    01   AE   0DA9  2387          MNEGW   #1,TTY$W_RB_CPZCUR(R4)     ; ELSE USE THE END POSITION
                     0DAD  2388 119$:    SET_STATE <CTRLR,EDITREAD>         ; INDICATE THAT WE ARE OUTPUTTING THE
  34 A4    3C A4   A1   0DB5  2389          ADDW3   TTY$W_RB_TXTOFF(R4),TTY$W_RB_PRMLEN(R4),-
            51          0DBA  2390                  R1                         ; GET THE NUMBER OF CHARACTERS TO OUTPUT
            07  13   0DBB  2391          BEQL    120$                       ; IF NONE THEN DON'T BOTHER.
```

```
                                                ;
                                                ; END OF CONTROL R
                                                ;
51  14 A4    D0  0DC4  2397 120$:    MOVL    TTY$L_RB_ECHSTR(R4),R1  ; CHECK FOR A CALLBACK
      1D     12  0DC8  2398          BNEQ    CALLBACK
                 0DCA  2399 MOVIFNEC:
    32 A4    B5  0DCA  2400          TSTW    TTY$W_RB_LINREST(R4)    ; ANY MORE CHARACTERS ON THE LINE
       03    13  0DCD  2401          BEQLU   EXITING                 ; NO THEN EXIT
     0083    31  0DCF  2402          BRW     MOVECURSOR              ; YES THEN MOVE THE CURSOR
    44 A4    B4  0DD2  2403 EXITING:CLRW     TTY$W_RB_MODE(R4)       ; CLEAN OUT THE EDITING MODE
    14 A4    D4  0DD5  2404          CLRL    TTY$L_RB_ECHSTR(R4)
                 0DD8  2405          CLR_STATE <EDITREAD,CTRLR,CTRLO,SKIPCRLF,TABEXPAND>
     F21E    31  0DE4  2406          BRW     GETNEXTCHAR
                 0DE7  2407 ;
                 0DE7  2408 ; IT MAY BE NECESSARY TO CALL THE EDITING CODE BACK TO DO SOME WORK OPERATION
                 0DE7  2409 ; THAT REQUIRED A ZERO ORIGIONAL CURSOR POSITION
                 0DE7  2410 ;
    44 A4    B4  0DE7  2411 CALLBACK:CLRW    TTY$W_RB_MODE(R4)       ; CLEAN OUT THE EDITING MODE
    14 A4    D4  0DEA  2412          CLRL    TTY$L_RB_ECHSTR(R4)
                 0DED  2413          CLR_STATE <EDITREAD,CTRLR,CTRLO,SKIPCRLF,TABEXPAND>
    3A A4    B5  0DF9  2414          TSTW    TTY$W_RB_CPZORG(R4)     ; MAKE SURE WE WON'T LOOP
       D4    12  0DFC  2415          BNEQ    EXITING                 ; IF NOT SUCCESSFULL IN CLEANING
                 0DFE  2416                                          ; THE ORIGION THEN DROP THE CHARACTER
       61    17  0DFE  2417          JMP     (R1)
```

```
                                                J 16
                     0E00  2419 ;
                     0E00  2420 ; ALTERNATE ECHO STRING.
                     0E00  2421 ;
44 A4    04   B0    0E00  2422 AESECHO: MOVW   #TTY$K_ER_EXITING,TTY$W_RB_MODE(R4); SETUP TO EXIT
51    28 A4   3C    0E04  2423          MOVZWL  TTY$W_RB_AESLEN(R4),R1  ; THE LENGTH OF THE STRING TO ECHO
54    24 A4   D0    0E08  2424          MOVL    TTY$L_RB_AES(R4),R4     ; AND THE ADDRESS
      026A    31    0E0C  2425          BRW     STRTMOLTI_1             ; NOW MULTIECHO
```

```
                                                K 16
                     0E0F  2427 ;
                     0E0F  2428 ; Echo and clear to end of the line
                     0E0F  2429 ;
                     0E0F  2430 CLRECHO:
                     0E0F  2431          SET_STATE <TABEXPAND>                ; SET TO EXPAND TABS
44 A4    06  9B      0E13  2432          MOVZBW  #TTY$K_ER_CLRREST,TTY$W_RB_MODE(R4); THEN MOVE THE CURSOR
51   0B A4  9A      0E17  2433 COMECHO:MOVZBL  TTY$B_RB_ECHLEN(R4),R1       ; SETUP THE NUNMBER OF CHARACTERS
54   14 A4  D0      0E1B  2434          MOVL    TTY$L_RB_ECHSTR(R4),R4       ; SETUP FOR MULTIECHO
     0257   31      0E1F  2435          BRW     STRTMULTI_1                  ; AND START THE MULTIECHOING
```

```
                                                L 16
                     0E22  2437 ;
                     0E22  2438 ; Clear to the end of the line
                     0E22  2439 ;
                     0E22  2440 CLRREST:
 44 A4    05   9B    0E22  2441          MOVZBW  #TTY$K_ER_MOVECURSOR,TTY$W_RB_MODE(R4);
    38 A4     B5    0E26  2442          TSTW    TTY$W_RB_CPZCUR(R4)     ; NEGITIVE CUR MEANS DON'T UPDATE
       06     18    0E29  2443          BGEQ    20$                     ;
 00FC C5      B0    0E2B  2444          MOVW    UCB$W_TT_CURSOR(R5),TTY$W_RB_CPZCUR(R4); UPDATE THE CURRENT
    38 A4           0E2F
 00FC C5      B1    0E31  2445 20$:     CMPW    UCB$W_TT_CURSOR(R5),UCB$W_DEVBUFSIZ(R5); ANYTHING TO CLEAR
    42 A5           0E35
       1C     13    0E37  2446          BEQL    MOVECURSOR              ; NOTHING TO CLEAR THEN JUST MOVE THE
                     0E39  2447                                          ; CURSOR
0A 48 A5     18    E1    0E39  2448          BBC     #TT2$V_ANSICRT,UCB$L_DEVDEPND2(R5),30$; IS IT AN ANSI CRT
  00000000'EF  9E    0E3E  2449          MOVAB   TTY$A_ANSICEL,R4        ; GET A CLEAR TO END OF LINE SEQUENCE
       54            0E44
     0223   31    0E45  2450          BRW     STRTMULTI               ; AND MULTIECHO IT OUT
                     0E48  2451 ; for non-ansi terminals
 44 A4    0B   9B    0E48  2452 30$:     MOVZBW  #TTY$K_ER_SIMCEOL,TTY$W_RB_MODE(R4); TELL IT TO COME BACK AND SIMULA
     0075   31    0E4C  2453          BRW     SIMCEOL
```

```
                        0E4F  2455 ;
                        0E4F  2456 ; echo the given data then exit
                        0E4F  2457 ;
 44 A4    04    9B      0E4F  2458 ECHLINE:MOVZBW  #TTY$K_ER_EXITING,TTY$W_RB_MODE(R4); EXIT AFTER ECHOING
          C2    11      0E53  2459          BRB     COMECHO                            ; AND GO TO THE COMMON CODE
```

```
                     0E55  2461 ;
                     0E55  2462 ; cause the cursor to be moved to it's origional position
                     0E55  2463 ;
                     0E55  2464 MOVECURSOR:
44 A4    04   9B    0E55  2465          MOVZBW  #TTY$K_ER_EXITING,TTY$W_RB_MODE(R4); SETUP TO EXIT
         0258 31    0E59  2466          BRW     UPDATE_CURSOR           ; AND UPDATE THE CURSOR
```

```
                     0E5C  2468 ;
                     0E5C  2469 ; Echo a prompt and a given amount of data then move the cursor to
                     0E5C  2470 ; the specifyed position.  This handles the no prompt case correctly.
                     0E5C  2471 ;
                     0E5C  2472 PRMECHO:
51    34 A4    B0    0E5C  2473          MOVW    TTY$W_RB_PRMLEN(R4),R1  ; GET THE LENGTH OF THE PROMPT
         11    13    0E60  2474          BEQL    40$                     ; NO PROMPT THEN JUST ECHO SPECIFYED STRING
44 A4    08    9B    0E62  2475          MOVZBW  #TTY$K_ER_PRMECHO1,TTY$W_RB_MODE(R4); SET THE NEXT STATE
54    4A A4    9E    0E66  2476          MOVAB   TTY$A_RB_PRMPT(R4),R4   ; GET THE PROMPT'S ADDRESS
                     0E6A  2477          SET_STATE <SKIPCRLF,SKIPLF>     ; SKIP THE FREE LINEFEEDS
       0206    31    0E70  2478          BRW     STRTMULTI_1             ; AND MULTIECHO THE PROMPT OUT
                     0E73  2479 ;
                     0E73  2480 ; SETUP THE CORRECT CURSOR POSITION FOR THE NO PROMPT CASE
                     0E73  2481 ;
     3A A4    B0    0E73  2482 40$:    MOVW    TTY$W_RB_CRZORG(R4),UCB$W_TT_CURSOR(R5)
   00FC C5         0E76
```

```
                     0E79  2484 ;
                     0E79  2485 ; second part of prompt echoing if necessary.
                     0E79  2486 ;
                     0E79  2487 PRMECHO1:
 38 A4    B5         0E79  2488          TSTW    TTY$W_RB_CPZCUR(R4)      ; SHOULD WE UPDATE THIS CURSOR
    0B    14         0E7C  2489          BGTR    50$                      ; NO IF NOT ZERO
 38 A4    B6         0E7E  2490          INCW    TTY$W_RB_CPZCUR(R4)      ; WAS THIS -1?
    06    19         0E81  2491          BLSS    50$                      ; NO THEN DON'T RESET THE CURSOR
00FC C5    B0         0E83  2492          MOVW    UCB$W_TT_CURSOR(R5),TTY$W_RB_CPZCUR(R4); RESET CURSOR POSITION
 38 A4               0E87
                     0E89  2493 50$:     CLR_STATE <SKIPCRLF,SKIPLF>      ; CLEAN THE SKIP STATE UP
 0B A4    95         0E8F  2494          TSTB    TTY$B_RB_ECHLEN(R4)      ; DO WE HAVE ANYTHING TO ECHO
    0B    14         0E92  2495          BGTR    70$                      ; YES THEN ECHO THE SPECIFYED STRING
    03    19         0E94  2496          BLSS    60$                      ; LESS THAN 0 EXIT
  FF89    31         0E96  2497          BRW     CLRREST                  ; ELSE THEN CLEAR THE REST OF THE LINE
 0B A4    94         0E99  2498 60$:     CLRB    TTY$B_RB_ECHLEN(R4)      ; CLEAN OUT THEE OFFSET JUST IN CASE
  FF33    31         0E9C  2499          BRW     EXITING
  FF6D    31         0E9F  2500 70$:     BRW     CLRECHO
```

```
                     0EA2  2502 ;
                     0EA2  2503 ; read verify echoing (echo's a string replacing fill characters with clear characte
                     0EA2  2504 ;
      0B A4    97    0EA2  2505 RVECHO: DECB    TTY$B_RB_ECHLEN(R4)          ; TAKE THIS CHARACTER OUT
         1A    19    0EA5  2506         BLSS    20$                          ; NO MORE THEN MOVE THE CURSOR
   53 14 B4    9A    0EA7  2507         MOVZBL  @TTY$L_RB_ECHSTR(R4),R3      ; GET A CHARACTER
   64 14 A4    D1    0EAB  2508         CMPL    TTY$L_RB_ECHSTR(R4),TTY$L_RB_TXT(R4); IS THIS PAST THE PROMPT
         0A    19    0EAF  2509         BLSS    10$                          ; NO THEN DON'T CHANGE CHARACTERS
   53 47 A4    91    0EB1  2510         CMPB    TTY$B_RB_RVFFIL(R4),R3       ; IS THIS A FILL CHARACTER
         04    12    0EB5  2511         BNEQ    10$                          ; NO THEN  CONTINUE
   53 46 A4    9A    0EB7  2512         MOVZBL  TTY$B_RB_RVFCLR(R4),R3       ; OTHERWISE ECHO THE CLEAR CHARACTER
      14 A4    D6    0EBB  2513 10$:    INCL    TTY$L_RB_ECHSTR(R4)          ; UPDATE THE COUNT
       F2CD    31    0EBE  2514         BRW     FORMAT_CHAR                  ; ELSE FORMAT THE CHARACTER
       FF06    31    0EC1  2515 20$:    BRW     MOVIFNEC                     ; MOVE IF NECESSARY
                     0EC4  2516
```

```
                        0EC4  2518 ;
                        0EC4  2519 ; NON-ANSI CRT THEN SIMULATE CLEAR TO END OF LINE
                        0EC4  2520 ;
                        0EC4  2521 $IMCEOL:
          0C    E1      0EC4  2522          BBC     #TT$V_SCOPE,-           ; CLEAR TO END OF LINE IS MEANINGLESS
     8C 44 A5            0EC6  2523                  UCB$L_DEVDEPEND(R5),MOVECURSOR; ON HARD COPY TERMINALS
      00FC C5    A3      0EC9  2524          SUBW3   UCB$W_TT_CURSOR(R5),UCB$W_DEVBUFSIZ(R5),R3
   53    42 A5            0ECD
      08    53    B1      0ED0  2525          CMPW    R3,#8                   ; OUTPUT IN GROUPS OF 8
            0A    15      0ED3  2526          BLEQ    10$                     ;...
 00000000'EF      9E      0ED5  2527          MOVAB   TTY$A_TAB,R4            ; SETUP TO ECHO 8
            54            0EDB
          018C    31      0EDC  2528          BRW     STRTMULTI
                        0EDF  2529
    44 A4   05    9B      0EDF  2530 10$:     MOVZBW  #TTY$K_ER_MOVECURSOR,TTY$W_RB_MODE(R4); NEXT MOVE THE CURSOR
51    53    01    C3      0EE3  2531          SUBL3   #1,R3,R1                ; SUBTRACT 1 FROM THE COUNT
 00000001'EF      9E      0EE7  2532          MOVAB   TTY$A_TAB+1,R4          ;.
            54            0EED
          0188    31      0EEE  2533          BRW     STRTMULTI_1             ; AND ECHO IT
```

```
                        OEF1  2535 ;
                        OEF1  2536 ; set the current cursor position
                        OEF1  2537 ;
                        OEF1  2538 UPDCURSOR:
     38 A4      B0      OEF1  2539          MOVW    TTY$W_RB_CPZCUR(R4),UCB$W_TT_CURSOR(R5): UPDATE THE CURSOR POSITION
00FC C5                 OEF4
       FED8     31      OEF7  2540          BRW     EXITING                 ; THEN EXIT
```

```
                                                H 1
                     0EFA  2542          .SBTTL  Read service routines
                     0EFA  2543 ;++
                     0EFA  2544 ; BACK_TAB
                     0EFA  2545 ;
                     0EFA  2546 ;       Calculates the cursor postion of the beginning of a tab while
                     0EFA  2547 ; at the end of the tab.
                     0EFA  2548 ;
                     0EFA  2549 ; Implicit inputs:
                     0EFA  2550 ;
                     0EFA  2551 ;       R5      UCB address
                     0EFA  2552 ;       TTY$W_RB_LINOFF the current location off of LIN
                     0EFA  2553 ;       TTY$L_RB_LIN the address of the beginning of this line
                     0EFA  2554 ;       TTY$L_RB_TXT the address of the beginning of the data buffer
                     0EFA  2555 ;       TTY$A_RB_PRM offset from svapte to the beginning of the prompt
                     0EFA  2556 ;       TTY$W_RB_CPZORG cursor postion when the read started
                     0EFA  2557 ;
                     0EFA  2558 ; Outputs:
                     0EFA  2559 ;       R3      Mod 8 of the cursor position
                     0EFA  2560 ;--
                     0EFA  2561 BACK_TAB:
          54    DD   0EFA  2562          PUSHL   R4                      ; SAVE R4
    54    78 A5   D0   0EFC  2563          MOVL    UCB$L_SVAPTE(R5),R4     ; GET THE ADDRESS OF THE READ BUFFER
    53    30 A4   3C   0F00  2564          MOVZWL  TTY$W_RB_LINOFF(R4),R3  ; GET THE CURRENT LOCATION
53     2C B443   9E   0F04  2565          MOVAB   @TTY$L_RB_LIN(R4)[R3],R3; POINT TO END OF DATA
          7E   D4   0F09  2566          CLRL    -(SP)                   ; SET UP CURSOR COUNTER
    64    2C A4   D1   0F0B  2567          CMPL    TTY$L_RB_LIN(R4),TTY$L_RB_TXT(R4); ARE WE ON THE FIRST LINE?
          06   13   0F0F  2568          BEQL    10$                     ; YES THEN CONTINUE
    54    2C A4   D0   0F11  2569          MOVL    TTY$L_RB_LIN(R4),R4     ; ELSE JUST USE THE BEGINNING OF THIS ONE
          04   11   0F15  2570          BRB     20$                     ; AND CONTINUE ON
    54    4A A4   9E   0F17  2571 10$:     MOVAB   TTY$L_RB_DATA(R4),R4    ; POINT TO DATA START
       54    53   D1   0F1B  2572 20$:     CMPL    R3,R4                   ; BUFFER START?
          23   13   0F1E  2573          BEQL    40$                     ; THEN REFERENCE POINT FOUND
       0D    73   91   0F20  2574          CMPB    -(R3),#TTY$C_CR         ; CARRIAGE RETURN?
          2C   13   0F23  2575          BEQL    50$                     ; IF EQL THEN REFERENCE POINT FOUND
       09    63   91   0F25  2576          CMPB    (R3),#TTY$C_TAB         ; TAB?
          27   13   0F28  2577          BEQL    50$                     ; IF EQL THEN REFERENCE POINT FOUND
       1B    63   91   0F2A  2578          CMPB    (R3),#TTY$C_ESCAPE      ; IS THIS AN ESCAPE
          0F   13   0F2D  2579          BEQL    30$                     ; YES THEN REMOVE THE ESCAPE SEQUENCE
9B     8F    63   91   0F2F  2580          CMPB    (R3),#TTY$C_CSI         ; CSI'S COUNT ALSO
          09   13   0F33  2581          BEQL    30$
       20    63   91   0F35  2582          CMPB    (R3),#TTY$C_BLANK       ; CURSOR CHANGE CHARACTER?
          E1   1F   0F38  2583          BLSSU   20$                     ; IF LSSU THEN NO
          6E   D6   0F3A  2584          INCL    (SP)                    ; ADJUST FAKE CURSOR
          DD   11   0F3C  2585          BRB     20$                     ; CONTINUE
                    0F3E  2586 ;
                    0F3E  2587 ; ESCAPE INTRODUCER FOUND
                    0F3E  2588 ;
        005C    30  0F3E  2589 30$:     BSBW    ESCAPE_TAB              ; REMOVE THE ESCAPE SEQUENCE
          D8    11  0F41  2590          BRB     20$                     ; THEN CONTINUE
                    0F43  2591 ;
                    0F43  2592 ; RAN OUT OF BUFFER WITHOUT FINDING A REFERENCE POINT
                    0F43  2593 ;
53     78 A5   D0   0F43  2594 40$:     MOVL    UCB$L_SVAPTE(R5),R3     ; GET THE ADDRESS OF THE READ BUFFER
2C A3     63   D1   0F47  2595          CMPL    TTY$L_RB_TXT(R3),TTY$L_RB_LIN(R3); THIS IS NOT THE FIRST LINE
          04   12   0F4B  2596          BNEQ    50$                     ; USE ZERO BASED ORIGION.
6E     3A A3   A0   0F4D  2597          ADDW    TTY$W_RB_CPZORG(R3),(SP); ELSE USE ORITIONAL ORIGION
                    0F51  2598 ;
```

```
                                                    I  1
                     0F51  2599 ; FOUND THE REFERENCE POINT THEN CONTINUE
                     0F51  260C ;
FFFFFFF8 8F    CB    0F51  2601 50$:    BICL3   #^XFFFFFFF8,(SP)+,R3    ; GET MOD 8 OF CURSOR
      53 8E          0F57
         54 8ED0     0F59  2602         POPL    R4                      ; RESTORE R4
            05       0F5C  2603         RSB
                     0F5D  2604
```

```
                     0F5D  2606 ;++
                     0F5D  2607 ; CHAR_SIZE
                     0F5D  2608 ;
                     0F5D  2609 ; DESCRIPTION:
                     0F5D  2610 ;
                     0F5D  2611 ;       GIVEN A CHARACTER IN R3 THIS ROUTINE WILL DETERMINE THE NUMBER
                     0F5D  2612 ; OF CURSOR POSITIONS THE CHARACTER WILL TAKE UP.
                     0F5D  2613 ;
                     0F5D  2614 ; INPUTS:
                     0F5D  2615 ;       R1 = 0 FOR FOWARD CHARACTER SIZE -1 FOR BACKWARD CHARACTER SIZE
                     0F5D  2616 ;       R3 = CHARACTER
                     0F5D  2617 ;       R5 = ADDRESS OF THE UCB
                     0F5D  2618 ;
                     0F5D  2619 ; OUTPUTS:
                     0F5D  2620 ;
                     0F5D  2621 ;       R3 = THE LENGTH IN CURSOR POSITIONS OF THIS CHARACTER
                     0F5D  2622 ;       R5 = UCB ADDRESS
                     0F5D  2623 ;
                     0F5D  2624 ;--
                     0F5D  2625
                     0F5D  2626 CHAR_SIZE:
                     0F5D  2627         CASE    W^TTY$A_TYPE[R3],LIMIT=#1@TTY$V_CH_SPEC,TYPE=B,<-
                     0F5D  2628                         10$,-                          ; Back space
                     0F5D  2629                         20$,-                          ; tab
                     0F5D  2630                         5$,-                           ; linefeed (no action)
                     0F5D  2631                         5$,-                           ; Vtab (no action)
                     0F5D  2632                         5$,-                           ; form (no action)
                     0F5D  2633                         60$>                           ; <CR>
      F0 8F    93    0F70  2634         BITB    #<TTY$M_CH_CTRL!TTY$M_CH_SPEC!TTY$M_CH_CTRL2!TTY$M_CH_CTRL3>,-
0000'CF43            0F73  2635         W^TTY$A_TYPE[R3]                       ; TEST FOR SPECIAL
            04    12 0F77  2636         BNEQ    5$                             ; NON SPACING CHARACTER
      53    01    9A 0F79  2637         MOVZBL  #1,R3                          ; OTHERWISE ONLY 1 CHARACTER POSITION
                  05 0F7C  2638         RSB
                     0F7D  2639
            53    D4 0F7D  2640 5$:     CLRL    R3                             ; CLEAR R3 OUT
                  05 0F7F  2641         RSB                                    ; AND RETURN
                     0F80  2642
      53    01    CE 0F80  2643 10$:    MNEGL   #1,R3                          ; BACKSPACE THEN BACKUP
                  05 0F83  2644         RSB
                     0F84  2645
                     0F84  2646 20$:                                          ; TAB
      03    00    EF 0F84  2647         EXTZV   #0,#3,UCB$W_TT_CURSOR(R5),R3; GET HORIZONTAL POINTER
53    00FC  C5           0F87
            51    D5 0F8B  2648         TSTL    R1                             ; IS THIS A BACKWARD CALCULATION
            03    18 0F8D  2649         BGEQ    23$                            ; NO THEN SUBTRACT
         FF68     30 0F8F  2650         BSBW    BACK_TAB                       ; CALCULATE THE MOD 8 OF THE CURSOR
53    08    53    C3 0F92  2651 23$:    SUBL3   R3,#8,R3                       ; NORMALIZE THE RESULT
                  05 0F96  2652         RSB                                    ; AND RETURN
                     0F97  2653
53    00FC  C5    3C 0F97  2654 60$:    MOVZWL  UCB$W_TT_CURSOR(R5),R3         ; CARRIAGE RETURN SHOULD ZERO CURSOR
                  05 0F9C  2655         RSB                                    ; POSITION
```

```
                        ;++
                        ; ESCAPE_TAB
                        ;
                        ;       Handle the removal of escape sequencses from a prompt
                        ; when deleting tabs.
                        ;
                        ; Implicit inputs:
                        ;
                        ;       R3 the address of the escape character
                        ;       R5 THE UCB ADDRESS
                        ;       4(SP) the internal count
                        ;
                        ; Implicit outputs:
                        ;
                        ;       4(SP) updated internal count
                        ;
                        ; All registers are saved
                        ;--
                        ESCAPE_TAB:
          51    DD      PUSHL   R1                      ; SAVE A SCRATCH REGISTER
    51    08 AE   D0      MOVL    8(SP),R1                ; RESTORE THE INTERNAL COUNT
          18    BB      PUSHR   #^M<R3,R4>              ; SAVE THE REST OF THE REGISTERS
       54 53    D0      MOVL    R3,R4                   ; MOVE THE CURRENT POSITION TO R4
       53 84    9A      MOVZBL  (R4)+,R3                ; GET THE CHARACTER THAT INTRODUCES THE SEQUENCE
          51    DD      PUSHL   R1                              ; SAVE R1
        F4F3    30      BSBW    ESCINIT                         ; INIT THE ESCAPE SEQUENCE RULES
0103 C5   51    90      MOVB    R1,UCB$B_TT_ESC(R5)             ; ...
          51 8ED0       POPL    R1                              ; RESTORE R1
                        ;
          51    D7      DECL    R1                      ; DECREMENT THE COUNT
       53 84    9A      MOVZBL  (R4)+,R3                ; GET THE CHARACTER AND MOVE TO THE NEXT
        F4F4    30      BSBW    ESCSYNTAX               ; CHECK THE SYNTAX
          F6    14      BGTR    20$                     ; CONTINUE IF NECESSARY
          18    BA      POPR    #^M<R3,R4>              ; SEQUENCE COMPLETE THEN RESTORE THE REGISTERS
08 AE     51    D0      MOVL    R1,8(SP)                ; PUT THE COUNT BACK
          51 8ED0       POPL    R1                      ; RESTORE THE LAST REGISTER
                05      RSB                             ; AND NOW RETURN
```

```
                                                     L 1
                     OFCC  2695 ;++
                     OFCC  2696 ; FIND_BOL - FIND THE BEGINNING OF THIS LINE
                     OFCC  2697 ;
                     OFCC  2698 ; Description:
                     OFCC  2699 ;
                     OFCC  2700 ;       Given a string this routine will find the offset to the character
                     OFCC  2701 ; that will end up in the first character pos'tion of the bottom line
                     OFCC  2702 ; of the screen
                     OFCC  2703 ;
                     OFCC  2704 ;
                     OFCC  2705 ; IMPLICIT INPUTS:
                     OFCC  2706 ;
                     OFCC  2707 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                     OFCC  2708 ;       R4 = ADDRESS OF THE READ BUFFER
                     OFCC  2709 ;       R5 = ADDRESS OF THE UCB
                     OFCC  2710 ;       TTY$L_RB_TXT
                     OFCC  2711 ;       TTY$L_RB_LIN
                     OFCC  2712 ;       TTY$L_RB_PRMLEN
                     OFCC  2713 ;       TTY$L_RB_TXTOFF assumed non-zero
                     OFCC  2714 ;       TTY$A_RB_PRM
                     OFCC  2715 ;       TTY$W_RB_LINOFF
                     OFCC  2716 ;       TTY$W_RB_LINREST assumed zero
                     OFCC  2717 ;
                     OFCC  2718 ; IMPLICIT OUTPUTS:
                     OFCC  2719 ;
                     OFCC  2720 ;       TTY$L_RB_LIN address of the first character in this line of data
                     OFCC  2721 ;       TTY$L_RB_LINOFF offset from LIN to the end of the line
                     OFCC  2722 ;       R3 is destroyed.
                     OFCC  2723 ;--
                     OFCC  2724 FIND_BOL:
     00FC C5    B4   OFCC  2725          CLRW    UCB$W_TT_CURSOR(R5)     ; CLEAN THE CURSOR POSITION
                     OFDO  2726 FIND_BOL_NOCLEAR::
          51    D4   OFDO  2727          CLRL    R1                      ; CLEAN R1
  2C A4     64    DO   OFD2  2728          MOVL    TTY$L_RB_TXT(R4),TTY$L_RB_LIN(R4); START WITH THE BEGINNING
     00C0 8F    BB   OFD6  2729          PUSHR   #^M<R6,R7>              ; GET A FEW SCRATCH REGISTERS
          56    D4   OFDA  2730          CLRL    R6                      ; CLEAR R6, THE BEGINNING OFFSET
3C A4    34 A4   A1   OFDC  2731          ADDW3   TTY$W_RB_PRMLEN(R4),TTY$W_RB_TXTOFF(R4),R7; GET THE FNDING OFFSET
          57         OFE1
00 2A A4   00    E5   OFE2  2732          BBCC    #TTY$V_RS_WRAP,TTY$W_RB_RDSTATE(R4),210$; CLEAR THE WRAP STATE
     57    56    B1   OFE7  2733 210$:    CMPW    R6,R7                   ; ARE WE DONE?
          6D    18   OFEA  2734          BGEQ    240$                    ; YES THEN EXIT
 53   4A A446    9A   OFEC  2735          MOVZBL  TTY$A_RB_PRM(R4)[R6],R3 ; GET THE CHARACTER
          56    D6   OFF1  2736          INCL    R6                      ; MOVE TO THE NEXT CHARACTER
          04    91   OFF3  2737          CMPB    #TTY$K_ET_ESCAPE,TTY$A_CCLIST[R3]
00000000'EF43         OFF5
          3B    13   OFFB  2738          BEQL    250$                    ; HANDLE ESCAPE SEQUENCES
     53   OD    91   OFFD  2739          CMPB    #TTY$C_CR,R3            ; A CR ZEROS CURSOR
          2B    13   1000  2740          BEQL    230$                    ; SO DO SO
        FF58    30   1002  2741          BSBW    CHAR_SIZE               ; OTHER WISE GET THECURSOR POSITION
 00FC C5    53    A0   1005  2742          ADDW    R3,UCB$W_T_CURSOR(R5)   ; ADD IN THE POSITION
     00FC C5    B1   100A  2743          CMPW    UCB$W_TT_CURSOR(R5),UCB$W_DEVBUFSIZ(R5); IS THIS A WRAP
       42 A5         100E
          D5    19   1010  2744          BLSS    210$                    ; NO THEN GO ON
       42 A5    A2   1012  2745          SUBW    UCB$W_DEVBUFSIZ(R5),UCB$W_TT_CURSOR(R5); SUBTRACT OUT THE
     00FC C5         1015
                     1018  2746                                          ; LINE LENGTH
 53   4A A446    9E   1018  2747          MOVAB   TTY$A_RB_PRM(R4)[R6],R3 ; GET THE ADDRESS OF THE WRAP
```

```
00 2A A4    00    E2  101D  2748 220$:   BBSS    #TTY$V_RS_WRAP,TTY$W_RB_RDSTATE(R4),225$; SET UP THE WRAP CHARACTERI
      64    53    D1  1022  2749 225$:   CMPL    R3,TTY$L_RB_TXT(R4)     ; ARE WE PAST THE PROMPT?
            C0    19  1025  2750          BLSS    210$                    ; YES THEN CONTINUE
         2C A4    53  D0  1027  2751      MOVL    R3,TTY$L_RB_LIN(R4)     ; AND THIS IS THE LINE
            BA    11  102B  2752          BRB     210$                    ; THEN CONTINUE
                      102D  2753
53    4A A446     9E  102D  2754 230$:   MOVAB   TTY$A_RB_PRM(R4)[R6],R3 ; GET THE ADDRESS OF THE WRAP
      00FC C5     B4  1032  2755          CLRW    UCB$W_TT_CURSOR(R5)     ; CLEAR CURSOR ON A <CR>
            E5    11  1036  2756          BRB     220$                    ; NOW CONTINUE
                      1038  2757
                      1038  2758 ;
                      1038  2759 ; ESCAPE SEQUENCE INTRODUCER
                      1038  2760 ;
            51    DD  1038  2761 250$:   PUSHL   R1                      ; SAVE R1
         F466     30  103A  2762          BSBW    ESCINIT                 ; INIT THE SYNTAX RULES
0103 C5     51    90  103D  2763          MOVB    R1,UCB$B_TT_ESC(R5)     ;
            51 8ED0  1042  2764          POPL    R1                      ; RESTORE THE REGISTER
                      1045  2765 260$:
      57    56    B1  1045  2766          CMPW    R6,R7                   ; ARE WE FINISHED
            0F    18  1048  2767          BGEQ    240$                    ; YES THEN EXIT
53    4A A446     90  104A  2768          MOVB    TTY$A_RB_PRM(R4)[R6],R3 ; GET THE NXT CHARACTER
            56    D6  104F  2769          INCL    R6                      ; MOVE TO THE NEXT CHARACTER
         F460     30  1051  2770          BSBW    ESCSYNTAX               ; AND CHECK THE SYNTAX
            EF    14  1054  2771          BGTR    260$                    ; MORE THEN CONTINUE
         FF8E     31  1056  2772          BRW     210$                    ; NO MORE THEN GO BACK
                      1059  2773
53    4A A446     9E  1059  2774 240$:   MOVAB   TTY$A_RB_PRM(R4)[R6],R3; GET THE ADDRESS OF THIS CHARACTER
   53    2C A4     C2  105E  2775          SUBL    TTY$L_RB_LIN(R4),R3     ; GET THE OFFSET TO THE LAST CHARACTER
   30 A4    53     B0  1062  2776          MOVW    R3,TTY$W_RB_LINOFF(R4)  ; AND LEAVE IT there
      00C0 8F     BA  1066  2777          POPR    #^M<R6,R7>              ; RESTORE THE REGISTERS
                  05  106A  2778          RSB
```

```
                     106B  2780 ;++
                     106B  2781 ; STRTMULTI - START A MULTIECHO STRING GIVEN THE ADDRESS OF AN ASCIC STRING
                     106B  2782 ;
                     106B  2783 ; FUNCTIONAL DESCRIPTION:
                     106B  2784 ;
                     106B  2785 ; THIS ROUTINE STARTS THE SPECIFIED MULTIECHO STRING ON A UNIT.
                     106B  2786 ; THE COUNT OF THE STRING IS PLACED IN UCB$W_TT_MULTILEN, AND THE
                     106B  2787 ; ADDRESS OF THE FIRST CHARACTER IN THE STRING IS PLACED IN UCB$L_TT_MULTI.
                     106B  2788 ; THEN STATE MULTI IS SET AND MULTIECHO IS CALLED.
                     106B  2789 ;
                     106B  2790 ; STRTMULTI_1 - starts a multiecho string given the address and a length
                     106B  2791 ;               of a string (address in R4, length in R1)
                     106B  2792 ;
                     106B  2793 ; INPUTS
                     106B  2794 ;
                     106B  2795 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                     106B  2796 ;       R4 = ADDRESS OF A COUNTED MULTIECHO STRING (ASCIC)
                     106B  2797 ;       R5 = UCB ADDRESS
                     106B  2798 ;
                     106B  2799 ; OUTPUTS:
                     106B  2800 ;
                     106B  2801 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                     106B  2802 ;       R3 = FIRST CHARACTER
                     106B  2803 ;       R5 = UCB ADDRESS
                     106B  2804 ;       R4 DESTROYED
                     106B  2805 ;--
                     106B  2806 STRTMULTI::
          00DC C5    B0  106B  2807         MOVW    UCB$W_TT_MULTILEN(R5),UCB$W_TT_SMLTLEN(R5); SAVE LENGTH
          00DE C5        106F
00DC C5    84    9B  1072  2808         MOVZBW  (R4)+,UCB$W_TT_MULTILEN(R5)     ;GET THE LENGTH INTO AL[LENA
               0C  11  1077  2809         BRB     LOADADR
                   1079  2810 STRTMULTI_1::
          00DC C5    B0  1079  2811         MOVW    UCB$W_TT_MULTILEN(R5),UCB$W_TT_SMLTLEN(R5); SAVE LENGTH
          00DE C5        107D
00DC C5    51    B0  1080  2812         MOVW    R1,UCB$W_TT_MULTILEN(R5)        ; MOVE IN THE NEW LENGTH
                   1085  2813 LOADADR:
          00D8 C5    D0  1085  2814         MOVL    UCB$L_TT_MULTI(R5),UCB$L_TT_SMLT(R5); SAVE THE ADDRESS
          00E0 C5        1089
00D8 C5    54    D0  108C  2815         MOVL    R4,UCB$L_TT_MULTI(R5)           ;THEN THE ADDRESS INTO MULTI
                   1091  2816         SET_STATE <MULTI>                       ;NEXT SET MULTIECHO STATE
        F3CD    31  1095  2817         brw     multiechoing
```

```
                     1098  2819 ;++
                     1098  2820 ; TABRIGHT - CHECK FOR A TAB TO THE RIGHT OF THE CURSOR POSITION
                     1098  2821 ;
                     1098  2822 ; DESCRIPTION:
                     1098  2823 ;
                     1098  2824 ;       This routine will detect a tab character to the right of the cursor
                     1098  2825 ; position and set a flag indicating that insert mode is illegal.  This
                     1098  2826 ; prevents having to recalculate tab and wrap position for each character.
                     1098  2827 ;
                     1098  2828 ; Inputs:
                     1098  2829 ;       R4 - Read packet address
                     1098  2830 ;
                     1098  2831 ; Outputs:
                     1098  2832 ;
                     1098  2833 ;       TABRIGHT state bit set if a tab is to the right of the cursor
                     1098  2834 ;               reset if not.
                     1098  2835 ;       All registers preserved.
                     1098  2836 ;--
                     1098  2837 TABRIGHT:
          03   BB    1098  2838         PUSHR   #^M<R0,R1>                      ; SAVE SOME WORKING REGISTERS
                     109A  2839         CLR_STATE TABRIGHT                      ; CLEAR THIS STATE
51   30 A4   3C      109F  2840         MOVZWL  TTY$W_RB_LINOFF(R4),R1          ; GET THE OFFSET INTO THE STRING
32 A4   09   3A      10A3  2841         LOCC    #TTY$C_TAB,TTY$W_RB_LINREST(R4),@TTY$L_RB_LIN(R4)[R1];
   2C B441           10A7
                     10AA  2842                                                 ; TRY TO FIND A TAB
          05   13    10AA  2843         BEQL    10$                             ; DID WE FIND A TAB?
                     10AC  2844         SET_STATE TABRIGHT                      ; YES THEN SET THE STATE
          03   BA    10B1  2845 10$:    POPR    #^M<R0,R1>                      ; RESTORE THE REGISTERS
               05    10B3  2846         RSB                                     ; AND RETURN
                     10B4  2847
```

```
                                                                ;++
                                                                ; UPDATE_CURSOR
                                                                ;
                                                                ; DESCRIPTION:
                                                                ;
                                                                ;       WILL SETUP THE NECESSARY SEQUENCE OF EVENTS TO HAVE
                                                                ; THE CURSOR MVOVED TO THE PLACE SPECIFYED BY TTY$W_RB_CPZCUR
                                                                ;
                                                                ; INPUTS:
                                                                ;
                                                                ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                                                                ;       R5 = UCB ADDRESS
                                                                ;       TTY$W_RB_CPZCUR = THE FINAL CURSOR POSITION
                                                                ;       UCB$W_TT_CURSOR = THE CURRENT POSITION.
                                                                ;
                                                                ; OUTPUT:
                                                                ;
                                                                ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                                                                ;       R4 = ADDRESS OF THE READ PACKET
                                                                ;       R5 = UCB ADDRESS
                                                                ; R3 DESTROYED
                                                                ;--
                                                                UPDATE_CURSOR:
      54    78 A5    D0  10B4  2872                                  MOVL    UCB$L_SVAPTE(R5),R4             ; GET THE READ PACKET ADDRESS
            38 A4    A3  10B8  2873                                  SUBW3   TTY$W_RB_CPZCUR(R4),-           ; SUBTRACT THE FINAL CURSOR POSITION
         00FC C5          10BB  2874                                          UCB$W_TT_CURSOR(R5),-           ; FROM THE CURRENT CURSOR POSITION
         0100 C5          10BE  2875                                          UCB$W_TT_BSPLEN(R5)             ; AND ECHO THAT MANY BACKSPACES
         00FC C5    B1    10C1  2876                                  CMPW    UCB$W_TT_CURSOR(R5),-
           42 A5          10C5  2877                                          UCB$W_DEVBUFSIZ(R5)             ; ARE WE AT THE END OF THE LINE
              0A    1F    10C7  2878                                  BLSSU   5$                              ; NO THEN CONTINUE
         00FC C5    B7    10C9  2879                                  DECW    UCB$W_TT_CURSOR(R5)             ; KEEP THE POSITION CORRECT
         0100 C5    B7    10CD  2880                                  DECW    UCB$W_TT_BSPLEN(R5)             ; ELSE THE TERMINAL DOES A BACKSPACE
              4D    13    10D1  2881                                  BEQL    40$                             ; NO BACKSPACES THEN EXIT
                          10D3  2882 5$:
   45 48 A5    18    E1    10D3  2883                                  BBC     #TT2$V_ANSICRT,UCB$L_DEVDEPND2(R5),30$; NOT AN ANSI TERMINAL THEN BR
    0100 C5    08    B1    10D8  2884                                  CMPW    #8,UCB$W_TT_BSPLEN(R5)          ; CAN WE OPTOMIZE ON AN ANSI TERMINA
               3E    14    10DD  2885                                  BGTR    30$                             ; NO THEN OUTPUT BACKSPACES
               3F    BB    10DF  2886                                  PUSHR   #^M<R0,R1,R2,R3,R4,R5>          ; SAVE THE REGISTERS
      00000000'EF    9A    10E1  2887                                  MOVZBL  TTY$A_ANSIBACKUP,R1             ; GET THE LENGTH OF THE STRING
               51          10E7
               51    D6    10E8  2888                                  INCL    R1                              ; ADD IN THE COUNT TOO.
               51    28    10EA  2889                                  MOVC3   R1,TTY$A_ANSIBACKUP,TTY$Q_RB_ECHOAREA(R4); MOVE THE CHARACTER
      00000000'EF          10EC
            0C A4          10F1
                          10F3  2890                                                                          ; STRING IN TO PLACE
               3F    BA    10F3  2891                                  POPR    #^M<R0,R1,R2,R3,R4,R5>          ; RESTORE THE REGISTERS
               7E    D4    10F5  2892                                  CLRL    -(SP)                           ; GET ENOUGH ROOM FOR THE DIVIDEND
   7E    0100 C5    3C    10F7  2893                                  MOVZWL  UCB$W_TT_BSPLEN(R5),-(SP)       ;  AND PUT IT THE DIVISOR
     51    12 A4    9E    10FC  2894                                  MOVAB   TTY$Q_RB_ECHOAREA+6(R4),R1      ; GET THE ADDRESS OF THE FISRST BYTE
   6E    6E    0A    7B    1100  2895 10$:                            EDIV    #10,(SP),(SP),R3                ; GET THE FIRST DIGIT
               53          1104
         71    53    80    1105  2896                                  ADDB    R3,-(R1)                        ; PUT IT IN PLACE
               6E    D5    1108  2897                                  TSTL    (SP)                            ; DO WE HAVE ANY MORE TO DO
               F4    12    110A  2898                                  BNEQ    10$                             ; YES THEN DO SO
               8E    D5    110C  2899                                  TSTL    (SP)+                           ; CLEAN THE STACK
               8E    D5    110E  2900                                  TSTL    (SP)+                           ;
            38 A4    B0    1110  2901                                  MOVW    TTY$W_RB_CPZCUR(R4),UCB$W_TT_CURSOR(R5); UPDATE THE CURSOR POSITION
```

```
                                                D 2
                     1113
                     1116  2902 ;       MOVZBW  #TTY$K_ER_UPDCURSOR,TTY$W_RB_MODE(R4); SETUP THE EDITREAD STATE
                     1116  2903 ;       SET_STATE EDITREAD                             ; And setup the edit read state
54   0C A4   9E      1116  2904         MOVAB   TTY$Q_RB_ECHOAREA(R4),R4               ; GET THE ECHO AREA ADDRESS
     FF4E    31      111A  2905         BRW     STRTMULTI                              ; and start the multiechoing
                     111D  2906
                     111D  2907 30$:     SET_STATE <BACKSPACE>                        ; AND START MULTIECHOING BACKSPACES
     EEE2    31      1120  2908 40$:     BRW     GETNEXTCHAR                          ; GET THE NEXT CHARACTER
```

```
                        1123  2910 ;++
                        1123  2911 ;ZERO_CPZORG - HANDLE CASE WHERE WE NEED A FRESH LINE
                        1123  2912 ;
                        1123  2913 ; DESCRIPTION:
                        1123  2914 ;
                        1123  2915 ;       This routine will check and make sure that the origional cursor
                        1123  2916 ; position becomes zero.  This will then callback the routine that called it
                        1123  2917 ; and re-run the operation.
                        1123  2918 ;
                        1123  2919 ; Inputs:
                        1123  2920 ;
                        1123  2921 ;       R4 - READ BUFFER ADDRESS
                        1123  2922 ;       R5 - UCB ADDRESS
                        1123  2923 ;       (SP) - ADDRESS OF A ROUTINE TO CONTINUE WITH
                        1123  2924 ;--
                        1123  2925 ZERO_CPZORG:
06 2A A4    00    E0    1123  2926         BBS     #TTY$V_RS_WRAP,TTY$W_RB_RDSTATE(R4),50$; IF WE WRAPPED THEN
                        1128  2927                                                 ; WE MUST FIX UP THE IO
   2C A4    64    D1    1128  2928         CMPL    TTY$L_RB_TXT(R4),TTY$L_RB_LIN(R4); DID WE WRAP?
            07    13    112C  2929         BEQL    60$                             ; NO THEN NO SPECAIL ACTION
   14 A4    8E    D0    112E  2930 50$:     MOVL    (SP)+,TTY$L_RB_ECHSTR(R4); SETUP A CALLBACK
         F79E    31    1132  2931         BRW     CTRLR                           ; AND REFRESH THE IO
                 05    1135  2932 60$:     RSB
```

```
                     1136  2934          .sbttl  Read passall
                     1136  2935 ;++
                     1136  2936 ;
                     1136  2937 ; PASS ALL  AND NO FILTER HANDELER
                     1136  2938 ;
                     1136  2939 ; INPUTS:
                     1136  2940 ;       R2 = ADDRESS OF THE UNIT STATE VECTOR
                     1136  2941 ;       R3 = CHARACTER TO INSERT
                     1136  2942 ;       R4 = READ PACKET ADDRESS
                     1136  2943 ;       R5 = UCB ADDRESS
                     1136  2944 ;
                     1136  2945 ;--
                     1136  2946 PASSALL::
                     1136  2947
   51    3C A4   3C  1136  2948          MOVZWL  TTY$W_RB_TXTOFF(R4),R1  ; GET THE PLACE TO PUT THIS CHARACTER
00 B441    53   90  113A  2949          MOVB    R3,@TTY$L_RB_TXT(R4)[R1]; PUT THE CHARACTER AWAY
                     113F  2950          IF_STATE ESC,1800$              ; ESCAPE THE EXIT
                     1143  2951          IF_NOT_STATE PASALL,1900$       ; NOFORMAT THEN UPPER CASING MAY BE NECESSAR
                     1147  2952 1005$:   IF_STATE ESCAPE,1045$          ; IF ESCAPE MODE THEN EXIT
62 1C B4    53   E0  114B  2953 1040$:   BBS     R3,@TTY$L_RB_TERM(R4),1500$; IS THIS A TERMINATOR
      3C A4   B6  1150  2954          INCW    TTY$W_RB_TXTOFF(R4)     ; INCREMENT THE COUNT
40 A4 3C A4   B1  1153  2955 1010$:   CMPW    TTY$W_RB_TXTOFF(R4),TTY$W_RB_TXTSIZ(R4); ARE WE FINSIHED
         46   1E  1158  2956          BGEQU   1600$                   ; YES THEN FINISH
                     115A  2957
                     115A  2958 1030$:   IF_STATE <ESC>,1000$            ; IS THIS THIS ESCAPE
                     115E  2959          IF_NOT_STATE PRE,1025$          ; IF PRETYPEAHEAD THEN RETURN
         05  1162  2960 1020$:   RSB
                     1163  2961 1025$:   IF_NOT_STATE NOECHO,1027$
         00  E1  1167  2962          BBC     #TT2$V_LOCALECHO,-      ; LOCAL ECHO?
   10 48 A5        1169  2963                  UCB$L_DEVDEPND2(R5),1026$; NO THEN MUST BE NOECHO
         06  E0  116C  2964          BBS     #TRM$V_TM_NOECHO,-      ; NOECHO SPECIFYED
   0B 20 A4        116E  2965                  TTY$L_RB_MOD(R4),1026$  ; ON THE READ
      F07A   31  1171  2966          BRW     FORMAT_LOCAL            ; ELSE LOCAL ECHO IT
      F017   31  1174  2967 1027$:   BRW     FORMAT_CHAR             ; FORMAT THE CHARACTER
                     1177  2968 1000$:   IF_NOT_STATE PRE,1026$
         05  117B  2969          RSB
      EE86   31  117C  2970 1026$:   BRW     GETNEXTCHAR
                     117F  2971
      0065   31  117F  2972 1045$:   BRW     1400$
                     1182  2973 ; CONVERT LOWER TO UPPER IF NOFORMAT
                     1182  2974 ;
   51    58 A5   D0  1182  2975 1900$:   MOVL    UCB$L_IRP(R5),R1        ; GET THE IRP ADDRESS
BC 20 A1    08  E1  1186  2976          BBC     #IO$V_CVTLOW,IRP$W_FUNC(R1),1005$; BR IF CONVERT LOWER TO UPPER
0000'CF43   03  E1  118B  2977          BBC     #TTY$V_CH_LOWER,W^TTY$A_TYPE[R3],1005$: BR IF NOT LOWER CHARACTER
                B5  1191
      53    20  AA  1192  2978          BICW    #^X020,R3               ; CONVERT TO UPPER CASE
   51    3C A4   3C  1195  2979          MOVZWL  TTY$W_RB_TXTOFF(R4),R1  ; GET THE PLACE TO PUT THIS CHARACTER
00 B441    53   90  1199  2980          MOVB    R3,@TTY$L_RB_TXT(R4)[R1]; PUT THE CHARACTER AWAY
         A7  11  119E  2981          BRB     1005$
                     11A0  2982
   51    58 A5   D0  11A0  2983 1600$:   MOVL    UCB$L_IRP(R5),R1        ; GET THE IRP ADDRESS
      38 A1   B4  11A4  2984          CLRW    IRP$L_MEDIA(R1)         ; SET NO TERMINATOR
                     11A7  2985          SET_STATE EOL                   ; SIGNIFY END OF LINE
B2 04 A2    1A  E4  11AB  2986          BBSC    #TTY$V_ST_PRE,4(R2),1020$; CLEAR PRE IF NECESSARY AND EXIT
         A8  11  11B0  2987          BRB     1030$
                     11B2  2988
                     11B2  2989
```

```
                30 A4   3C A4    B0   11B2  2990 1500$:  MOVW    TTY$W_RB_TXTOFF(R4),TTY$W_RB_LINOFF(R4); SETUP LINOFF
                                     11B7  2991          SET_STATE EOL                       ; SIGNIFY END OF LINE
                          66    11   11BB  2992          BRB     1440$
                                     11BD  2993 ;
                                     11BD  2994 ; ESCAPE IN PROGRESS
                                     11BD  2995 ;
                     51   58 A5    D0   11BD  2996 1800$:  MOVL    UCB$L_IRP(R5),R1            ; GET THE IRP ADDRESS
                          3A A1    B6   11C1  2997          INCW    IRP$L_MEDIA+2(R1)           ; ADD 1 TO THE SIZE
                          3C A4    B6   11C4  2998          INCW    TTY$W_RB_TXTOFF(R4)         ; UP THE OFFSET BECAUSE WE PUT 1 CHAR IN THE
                           F2EA    30   11C7  2999          BSBW    ESCSYNTAX                   ; CHECK THE SYNTAX OF THIS CHAR
                             18    14   11CA  3000          BGTR    1830$                       ; OK THEN CONTINUE ON
                             04    13   11CC  3001          BEQL    1810$                       ; VALID SEQUENCE CONTINUE
                                     11CE  3002
                                     11CE  3003          SET_STATE BADESC                    ; A BAD ESCAPE SEQUENCE THEN SAY SO
                                     11D2  3004 1810$:   CLR_STATE ESC                       ; CLEAR THE ESCAPE STATE
                30 A4   3A A1    A2   11D7  3005          SUBW    IRP$L_MEDIA+2(R1),TTY$W_RB_TXTOFF(R4); SUBTRACT OUT THE SEQUENCE
                                     11DC  3006          SET_STATE <EOL,NOECHO>              ; LENGTH AND SET END OF LINE
                                     11E4  3007
                              0047   31   11E4  3008 1830$:  BRW     1450$                       ; THEN EXIT
                                     11E7  3009
                00000000'EF43        91   11E7  3010 1400$:  CMPB    TTY$A_CCLIST[R3],#TTY$K_ET_ESCAPE; CHECK FOR ESCAPE INTR
                                04   11EE
                                5C   12   11EF  3011          BNEQ    1470$                       ;ESCAPE SEQUENCE INTRODUCER
                     53   7D 8F    91   11F1  3012          CMPB    #TTY$C_LOWESC1,R3           ;IS THIS AN ALTMODE
                                06   13   11F5  3013          BEQL    1410$                       ; YES THEN HANDLE SPECAILLY
                     53   7E 8F    91   11F7  3014          CMPB    #TTY$C_LOWESC2,R3           ; LOWER ALSO
                                11   12   11FB  3015          BNEQ    1420$                       ; NO THEN MUST BE AN ESC OR CSI
                4B 44 A5        07   E0   11FD  3016 1410$:  BBS     #TT$V_LOWER,UCB$L_DEVDEPEND(R5),1470$; LOWER CASE TERMINAL
                   51   3C A4    3C   1202  3017          MOVZWL  TTY$W_RB_TXTOFF(R4),R1      ; GET THE LOCATION OF THIS CHARACTER
                 00 B441        1B   90   1206  3018          MOVB    #TTY$C_ESCAPE,@TTY$L_RB_TXT(R4)[R1]; AND FORCE THE ESCAPE
                      53        1B   9A   1208  3019          MOVZBL  #TTY$C_ESCAPE,R3            ; AND PUT THE ESCAPE IN R3
                          3C A4    B6   120E  3020 1420$:  INCW    TTY$W_RB_TXTOFF(R4)         ; ESCAPE INTRO THEN SAY SO.
                                51   DD   1211  3021          PUSHL   R1                          ; SAVE
                           F28D    30   1213  3022          BSBW    ESCINIT
                 0103 C5       51   90   1216  3023          MOVB    R1,UCB$B_TT_ESC(R5)
                                51 8ED0   121B  3024          POPL    R1
                                     121E  3025          SET_STATE ESC                       ; SETUP ESCAPE STATE
                   51   58 A5    D0   1223  3026 1440$:  MOVL    UCB$L_IRP(R5),R1            ; GET THE IRP ADDRESS
                38 A1       53    B0   1227  3027          MOVW    R3,IRP$L_MEDIA(R1)          ; SETUP THE TERMINATING CHARACTER
                          3A A1    B6   122B  3028          INCW    IRP$L_MEDIA+2(R1)           ; AND THE LENGTH
                40 A4   3C A4    B1   122E  3029 1450$:  CMPW    TTY$W_RB_TXTOFF(R4),TTY$W_RB_TXTSIZ(R4); ARE WE FINISHED
                                15   1E   1233  3030          BGEQU   1460$                       ; YES THEN FINISH
                                     1235  3031          IF_NOT_STATE EOL,1480$              ; IF NOT END OF LINE THEN RETURN
                12 04 A2        1A   E4   1239  3032          BBSC    #TTY$V_ST_PRE,4(R2),1490$
                                     123E  3033          IF_STATE NOECHO,1480$
                                0C   E0   1242  3034          BBS     #TRM$V_TM_TRMNOECHO,-
                     0C 20 A4         1244  3035                  TTY$L_RB_MOD(R4),1495$      ; IF NOT ECHOING TERMINATER THEN EXIT
                           FF10    31   1247  3036 1480$:  BRW     1030$
                           FF53    31   124A  3037 1460$:  BRW     1600$
                           FEFB    31   124D  3038 1470$:  BRW     1040$
                           FF0F    31   1250  3039 1490$:  BRW     1020$
                           FF26    31   1253  3040 1495$:  BRW     1026$
```

```
                     1256  3042          .SBTTL  READ WITH VERIFICATION
                     1256  3043 ;++
                     1256  3044 ; RDVERIFY
                     1256  3045 ;
                     1256  3046 ; Description:
                     1256  3047 ;       Read verify allows programs that wish to do character validataion
                     1256  3048 ; tc issue one IO rather than a QIO per character as was previously the case.
                     1256  3049 ;
                     1256  3050 ; Inputs:
                     1256  3051 ;
                     1256  3052 ;       R2 - Unit state vector
                     1256  3053 ;       R5 - address of the ucb
                     1256  3054 ;
                     1256  3055 ; Implicit inputs:
                     1256  3056 ;
                     1256  3057 ;       UCB$L_SVAPTE    - The address of the read buffer.
                     1256  3058 ;       TTY$L_RB_TXT    - The address of the first character of the initial string
                     1256  3059 ;       TTY$W_RB_TXTOFF - Offset to the last character in the initial string
                     1256  3060 ;       TTY$W_RB_TXTSIZ - Length of the data buffer.
                     1256  3061 ;       TTY$W_RB_LINOFF - Offset to the end of the field, initial offset.
                     1256  3062 ;       TTY$L_RB_PIC    - The address of the picture string.
                     1256  3063 ;--
                     1256  3064 RDVERIFY::
54   00E4 C5   D0    1256  3065          MOVL    UCB$L_TT_TYPAHD(R5),R4  ; ADDRESS TYPEAHEAD BUFFER
          25   13    125B  3066          BEQL    40$                     ; NO BUFFER THEN THERE IS A FORK
                     125D  3067                                          ; PROCESS WAITING SO LET IT GO.
       0C A4   B5    125D  3068          TSTW    TTY$W_TA_INAHD(R4)      ; ANY TYPEAHEAD DATA?
          25   12    1260  3069          BNEQ    50$                     ; yes then take it out and continue
                     1262  3070 ;
                     1262  3071 ; TYPEAHEAD BUFFER EXHAUSTED
                     1262  3072 ;
                     1262  3073          IF_NOT_STATE <TYPFUL>,35$       ; BR IF TYPEAHEAD IS NOT FULL
                     1266  3074
        F3D0   31    1266  3075 20$:     BRW     XON                     ; CONTINUE
                     1269  3076 35$:
                     1269  3077 ;
                     1269  3078 ; THE TYPEAHEAD BUFFER HAS NOW BEEN EMPTIED INTO THE SYSTEM BUFFER.
                     1269  3079 ; IF PROCESSING A READ WITH ZERO SECOND TIMEOUT, RETURN THE DATA
                     1269  3080 ; TO THE USER IMMEDIATELY.
                     1269  3081 ;
14 68 A5    01   E1  1269  3082          BBC     #UCB$V_TT_TIMO,UCB$W_DEVSTS(R5),40$; BR IF NOT READ WITH TIMEOUT.
  00000000'EF   D1    126E  3083          CMPL    TTY$A_MAXTIME,UCB$L_TT_RDUE(R5); ZERO SECOND TIMEOUT?
      0080 C5         1274
          09   12    1277  3084          BNEQ    40$                     ; BR IF NOT.
      022C 8F   B0    1279  3085          MOVW    #SS$_TIMEOUT,UCB$W_BOFF(R5); SET TIMEOUT COMPLETION STATUS.
        7C A5
         ED7E'  30    127F  3086          BSBW    TTY$READONE             ; COMPLETE REQUEST.
                     1282  3087 ;
                     1282  3088 ; DISMISS INTERRUPT
                     1282  3089 ;
      010B C5   94   1282  3090 40$:     CLRB    UCB$B_TT_OUTYPE(R5)     ; SET NO RETURN CHARACTER.
                05   1286  3091          RSB                             ; and dismiss the interupt
                     1287  3092 ;
                     1287  3093 ; ACTUALLY REMOVE CHARACTER FROM BUFFER
                     1287  3094 ;
   53   04 B4   9A   1287  3095 50$:     MOVZBL  @TTY$L_TA_GET(R4),R3    ; GET THE CHARACTER
0F 04 A2    19   E5  128B  3096          BBCC    #TTY$V_ST_ECHAES,4(R2),55$; IS THIS THE FIRST CHARACTER
```

```
                54    78 A5    D0  1290  3097          MOVL    UCB$L_SVAPTE(R5),R4      ; GET THE READ PACKET ADDRESS
            44 A4     09    9B  1294  3098          MOVZBW  #TTY$K_ER_AESECHO,TTY$W_RB_MODE(R4); YES THEN ECHO THE AES STRING
                            1298  3099          SET_STATE EDITREAD               ; NOW DO EDITING
                   FAB7     31  129C  3100          BRW     EDITREAD                 ; AND ECHO THE STRING
                            129F  3101 55$:
                   0C A4    B7  129F  3102          DECW    TTY$W_TA_INAHD(R4)       ; ADJUST NUMBER OF TYPEAHEAD CHARACTERS
                            12A2  3103 ;
                            12A2  3104 ; CHECK FOR BUFFER WRAP AROUND
                            12A2  3105 ;
            04 A4    10 A4    F2  12A2  3106          AOBLSS  TTY$L_TA_END(R4),TTY$L_TA_GET(R4),60$; POINTER PAST END?
                            06  12A7
                0118 C4     9E  12A8  3107          MOVAB   TTY$L_TA_DATA(R4),TTY$L_TA_GET(R4); RESET POINTER
                  04 A4         12AC
                            12AE  3108
                      50    DD  12AE  3109 60$:     PUSHL   R0                       ; charo can not destroy R0
                54    78 A5    D0  12B0  3110          MOVL    UCB$L_SVAPTE(R5),R4              ; GET READ BUFFER
                            12B4  3111          IF_STATE ESC,TO$                 ; IF ESCAPE SEQUENCE IN PROGRESS THEN HANDLE
            20 A4     11    E1  12B8  3112          BBC     #TRM$V_TM_R_JUST,TTY$L_RB_MOD(R4),-; BRANCH IF LEFT JUSTIFY MODE
                        06  12BC  3113                  LEFT_JUSTIFY
                   015C     31  12BD  3114          BRW     RIGHT_JUSTIFY
                   029C     31  12C0  3115 10$:     BRW     ESCAPE_TERM
```

```
                                12C3  3117 ;
                                12C3  3118 ; THIS ROUTINE HANDLES LEFT JUSTIFIED FIELDS
                                12C3  3119 ;
                                12C3  3120
                                12C3  3121 LEFT_JUSTIFY:
                                12C3  3122
      51    30 A4    3C        12C3  3123          MOVZWL  TTY$W_RB_LINOFF(R4),R1          ; GET CURRENT OFFSET INTO FIELD
                                12C7  3124 ;
                                12C7  3125 ;        TERMINATE IF ALREADY FIELD FULL - THIS CAN OCCUR ON A NON AUTOTAB
                                12C7  3126 ;        FIELD WHERE WE DEFER "FULL FIELD" COMPLETION UNTIL ONE MORE
                                12C7  3127 ;        CHARACTER IS TYPED. CHAR TYPED IS TERMINATOR.
                                12C7  3128 ;
      3C A4    51    B1        12C7  3129          CMPW    R1,TTY$W_RB_TXTOFF(R4)          ; FIELD FULL?
               03    19        12CB  3130          BLSS    5$                              ; IF LSS, NOT YET
             00A1    31        12CD  3131          BRW     TERM                            ; YES - INPUT CHAR IS TERM
                                12D0  3132 5$:
                                12D0  3133
                                12D0  3134 ; Check to see if this character is valid in this position
                                12D0  3135
      50    18 B441    9A        12D0  3136          MOVZBL  @TTY$L_RB_PIC(R4)[R1],R0        ; ADDRESS CORRESPONDING PIC VALUE
               50    93        12D5  3137          BITB    R0,VERIFY_ARRAY[R3]             ; CHAR VALID ?
 00000000'EF43                  12D7
               03    12        12DD  3138          BNEQ    10$                             ; YES,
             008F    31        12DF  3139          BRW     TERM                            ; ELSE TERMINATOR
                                12E2  3140 10$:
                                12E2  3141 ;
                                12E2  3142 ;        UPCASE IF REQUESTED.
                                12E2  3143 ;
   0A 20 A4    08    E1        12E2  3144          BBC     #TRM$V_TM_CVTLOW,TTY$L_RB_MOD(R4),13$; NO CONVERT THEN DON'T
 0000'CF43     03    E1        12E7  3145          BBC     #TTY$V_CH_LOWER,W^TTY$A_TYPE[R3],13$; is this lower case?
                   03    12ED
         53    20    8A        12EE  3146          BICB    #^X20,R3                        ; yes MAKE LOWERCASE UPPER
                                12F1  3147 13$:
      00 B441    53    90        12F1  3148          MOVB    R3,@TTY$L_RB_TXT(R4)[R1]        ; STORE INTO READ BUFFER
                                12F6  3149 20$:
               51    B6        12F6  3150          INCW    R1                              ; POINT TO NEXT POSITION
      3C A4    51    B1        12F8  3151          CMPW    R1,TTY$W_RB_TXTOFF(R4)          ; FIELD FULL?
               5D    18        12FC  3152          BGEQ    FULL_1                          ; YES, WITH SINGLE CHARACTER
                                12FE  3153
                                12FE  3154 ; Check for trailing marker characters to be skipped
                                12FE  3155
      18 B441    95        12FE  3156          TSTB    @TTY$L_RB_PIC(R4)[R1]           ; NEXT POSITION MARKER?
               39    12        1302  3157          BNEQ    OUTPUT_1                        ; NO, OUTPUT 1 CHARACTER
                                1304  3158
               51    D7        1304  3159          DECL    R1                              ; POINT TO PREVIOUS POSITION
      00 B441    9E        1306  3160          MOVAB   @TTY$L_RB_TXT(R4)[R1],-         ; SAVE ADDRESS OF STRING TO OUTPUT
         14 A4                  130A
                                130C  3161                  TTY$L_RB_ECHSTR(R4)
      50    01    9A        130C  3162          MOVZBL  #1,R0                           ; INIT COUNT OF CHARACTERS IN STRING
                                130F  3163 30$:
               51    D6        130F  3164          INCL    R1                              ; POINT TO NEXT POSITION
      3C A4    51    B1        1311  3165          CMPW    R1,TTY$W_RB_TXTOFF(R4)          ; FIELD FULL
               03    19        1315  3166          BLSS    35$                             ; NO
             0041    31        1317  3167          BRW     FULL_1                          ; YES, SO DON'T OUTPUT
                                131A  3168                                                  ; TRAILING MARKERS
                                131A  3169 35$:
      18 B441    95        131A  3170          TSTB    @TTY$L_RB_PIC(R4)[R1]           ; NEXT SLOT A MARKER
```

```
                04   12  131E  3171          BNEQ    45$                                     ; NO
                50   B6  1320  3172          INCW    R0                                      ; COUNT MARKERS
                EB   11  1322  3173          BRB     30$                                     ; CONTINUE LOOKING
                         1324  3174 45$:
0B A4           50   90  1324  3175          MOVB    R0,TTY$B_RB_ECHLEN(R4)                  ; SAVE STRING LENGTH
                         1328  3176 ;        BRW     OUTPUT_S                                ; OUTPUT STRING
                         1328  3177
                         1328  3178 OUTPUT_S:                                              ; OUTPUT A STRING
30 A4           51   B0  1328  3179          MOVW    R1,TTY$W_RB_LINOFF(R4)                  ; SAVE NEW OFFSET
                         132C  3180          IF_STATE NOECHO,DONE                            ; NOECHO THEN DON'T ECHO
      32 A4     B4   1330  3181          CLRW    TTY$W_RB_LINREST(R4)                    ; DON'T CHANGE CURSOR POSITION
44 A4   0A   9B   1333  3182          MOVZBW  #TTY$K_ER_RVECHO,TTY$W_RB_MODE(R4); SETUP TO ECHO
                         1337  3183          SET_STATE EDITREAD                              ;...
                18   11  133B  3184          BRB     DONE                                    ; THEN ALLOW THE CODE TO FLOW
                         133D  3185
                         133D  3186 OUTPUT_1:                                              ; OUTPUT CHARACTER IN R3
30 A4           51   B0  133D  3187          MOVW    R1,TTY$W_RB_LINOFF(R4)                  ; SAVE NEW OFFSET
47 A4           53   91  1341  3188          CMPB    R3,TTY$B_RB_RVFFIL(R4)                  ; USER TYPE FILLCHAR
                04   12  1345  3189          BNEQ    EXIT                                    ; NO
53   46 A4      90   1347  3190          MOVB    TTY$B_RB_RVFCLR(R4),R3                  ; YES - REP W/CLR CHAR
                         134B  3191
                         134B  3192 EXIT:
                         134B  3193          IF_STATE NOECHO,DONE                            ; NO EDCHO THEN DON'T ECHO
      50 8ED0   134F  3194          POPL    R0
    EE39   31   1352  3195          BRW     FORMAT_CHAR                             ; FORMAT THE CHARACTER ELSEWISE
                         1355  3196
                         1355  3197 DONE:
      50 8ED0   1355  3198          POPL    R0                                      ; RESTORE R0
    ECA5   31   1358  3199          BRW     TTY$GETNEXTCHAR                         ; AND GET THE NEXT CHARACTER
                         135B  3200 FULL_1:                                         ; FIELD FULL, OUTPUT SINGLE CHARACTE
                         135B  3201 ;
                         135B  3202 ;        IF NOT AUTOTAB, DONT END NOW - LET HIM HIT A CHAR
                         135B  3203 ;
                12   E1  135B  3204          BBC     #TRM$V_TM_AUTO_TAB,-
      20 A4             135D  3205                  TTY$L_RB_MOD(R4),-
                DD       135F  3206                  OUTPUT_1                        ; ECHO LAST CHAR
                         1360  3207
30 A4           51   B0  1360  3208          MOVW    R1,TTY$W_RB_LINOFF(R4)                  ; SAVE NEW OFFSET
                         1364  3209          SET_STATE EOL                                   ; TELL THEM WE ARE DONE
54   58 A5      D0   1368  3210          MOVL    UCB$L_IRP(R5),R4                        ; GET IRP ADDRESS
      38 A4     D4   136C  3211          CLRL    IRP$L_MEDIA(R4)                         ; SIGNAL NO TERMINATOR
                DA   136F  3212          BRB     EXIT
                         1371  3213
                         1371  3214 TERM:
30 A4           51   B0  1371  3215          MOVW    R1,TTY$W_RB_LINOFF(R4)                  ; SAVE ENDING OFFSET
                         1375  3216          IF_STATE NOFLTR,3$                              ; SKIP EDITING IF NOFLTR
                03   91  1379  3217          CMPB    #TTY$K_ET_DELEFT,TTY$A_CCLIST[R3]; IS THIS A DELETE CHARACTER
00000000'EF43            137B
                46   13  1381  3218          BEQL    DELEFT_JUST                             ; YES THEN DELETE
                         1383  3219 3$:       IF_NOT_STATE ESCAPE,4$                          ; ESCAPE PROCESSING THEN
                04   91  1387  3220          CMPB    #TTY$K_ET_ESCAPE,TTY$A_CCLIST[R3]; ESCAPE ?
00000000'EF43            1389
                24   13  138F  3221          BEQL    10$                                     ; YES
                         1391  3222 4$:       SET_STATE EOL                                   ; SET END OF LINE
50   58 A5      D0   1395  3223 5$:       MOVL    UCB$L_IRP(R5),R0
38 A0           53   9B  1399  3224          MOVZBW  R3,IRP$L_MEDIA(R0)                      ; TERMINATOR
3A A0           01   90  139D  3225          MOVB    #1,IRP$L_MEDIA+2(R0)                    ; TERM LENGTH
```

```
                3B A0    51    90  13A1  3226          MOVB    R1,IRP$L_MEDIA+3(R0)            ; OFFSET TO INVALID CHAR
                51    3C A4    3C  13A5  3227          MOVZWL  TTY$W_RB_TXTOFF(R4),R1          ; GET THE OFFSET TO THE END
                                   13A9  3228                                                  ; OF THE BUFFER
                   00 B441    9E  13A9  3229          MOVAB   @TTY$L_RB_TXT(R4)[R1],TTY$L_RB_PIC(R4);SETUP THE END OF THE BUFFER
                      18 A4        13AD
                18 B4    53    90  13AF  3230          MOVB    R3,@TTY$L_RB_PIC(R4)            ; STORE IN BUFFER
                         A0    11  13B3  3231          BRB     DONE
                                   13B5  3232 ;
                                   13B5  3233 ; SIGNAL ESCAPE SEQUENCE ACTIVE
                                   13B5  3234 ;
                                   13B5  3235 10$:
                                   13B5  3236          SET_STATE ESC
                         51    DD  13BA  3237          PUSHL   R1                      ; SAVE R1
                      FOE4    30  13BC  3238          BSBW    ESCINIT                 ; INIT THE ESCAPE SEQUENCE RULES
                0103 C5    51    90  13BF  3239          MOVB    R1,UCB$B_TT_ESC(R5)     ; ...
                         51 8ED0  13C4  3240          POPL    R1                      ; RESTORE R1
                                   13C7  3241
                         CC    11  13C7  3242          BRB     5$                              ; SKIP SETTING END
                                   13C9  3243 ;
                                   13C9  3244 ; DELETE ONE CHARACTER TO THE LEFT
                                   13C9  3245 ;
                                   13C9  3246 ;       R1 = offset into the line
                                   13C9  3247 ;
                                   13C9  3248 DELEFT_JUST:
                         50    D4  13C9  3249          CLRL    R0                              ; A COUNTER
                         51    D7  13CB  3250          DECL    R1                              ; move left 1 character
                         0C    19  13CD  3251          BLSS    100$                            ; NO CHARACTER TO DELETE THEN DON'T
                         50    D6  13CF  3252 10$:     INCL    R0                              ; MOVE BACK 1 CHARACTER
                   18 B441    95  13D1  3253          TSTB    @TTY$L_RB_PIC(R4)[R1]           ; IS THIS A MARKER?
                         07    12  13D5  3254          BNEQ    20$                             ; NO THEN SHIFT
                         51    D7  13D7  3255          DECL    R1                              ; ELSE MOVE OVER IT
                         F4    18  13D9  3256          BGEQ    10$                             ; NO NON-MARKER THEN IGNORE THE DEL
                      FF77    31  13DB  3257 100$:    BRW     DONE                            ; ...
                                   13DE  3258
                3C A4    30 A4    B1  13DE  3259 20$:     CMPW    TTY$W_RB_LINOFF(R4),TTY$W_RB_TXTOFF(R4); WERE WE ABOUT TO TERMINATE
                            03    12  13E3  3260          BNEQ    25$                             ; NO THEN PROCESS NORMALY
                         50    01  D0  13E5  3261          MOVL    #1,R0                           ; ELSE ONLY GO BACK 1 PLACE
                                   13E8  3262                                                  ; DUE TO SPECAIL EOF CONDITIONS
                      30 A4    51  B0  13E8  3263 25$:     MOVW    R1,TTY$W_RB_LINOFF(R4)          ; UPDATE THE OFFSET
                                   13EC  3264          IF_STATE NOECHO,50$
                00FC C5    50    A3  13F0  3265          SUBW3   R0,UCB$W_TT_CURSOR(R5),TTY$W_RB_CPZCUR(R4); SET THE FINAL CURSOR
                         38 A4        13F5
                0100 C5    50    B0  13F7  3266          MOVW    R0,UCB$W_TT_BSPLEN(R5)          ; AND TELL HOW MANY CHARACTERS TO
                                   13FC  3267                                                  ; BACKSPACE OVER
                                   13FC  3268          SET_STATE <BACKSPACE,EDITREAD>          ; THEN OUTPUT OTHER STUFF AND RETUR"
                         00 B441    9E  1401  3269          MOVAB   @TTY$L_RB_TXT(R4)[R1],TTY$L_RB_ECHSTR(R4); GET THE ADDRESS OF WHERE
                            14 A4        1405
                   0B A4    01    90  1407  3270          MOVB    #1,TTY$B_RB_ECHLEN(R4)          ; THE LENGTH OF THE STRING TO OUTPUT
                   44 A4    0A    98  140B  3271          MOVZBW  #TTY$K_ER_RVECHO,TTY$W_RB_MODE(R4); SET THE MODE OF THE EDITING
                   32 A4    01    D0  140F  3272          MOVL    #1,TTY$W_RB_LINREST(R4)         ; TELL THE MOVE TO RESTORE THE POSIT
                                   1413  3273
                         47 A4    90  1413  3274 50$:     MOVB    TTY$B_RB_RVFFIL(R4),@TTY$L_RB_TXT(R4)[R1]; AND FILL THE BLANK
                         00 B441        1416
                                   1419  3275                                                  ; WITH THE FILL CHARACTER
                            FF39    31  1419  3276          BRW     DONE
```

```
                     141C  3278 RIGHT_JUSTIFY:
                     141C  3279
     51    30 A4     32  141C  3280          CVTWL   TTY$W_RB_LINOFF(R4),R1          ; GET CURRENT OFFSET
           03        18  1420  3281          BGEQ    3$                             ; IF WE ARE BEOND THE
                     1422  3282                                                     ; BOUNDARY THEN TERMINATE
           00BD      31  1422  3283          BRW     RIGHT_TERM
                     1425  3284 3$:
     50    18 B441   9A  1425  3285          MOVZBL  @TTY$L_RB_PIC(R4)[R1],R0       ; GET PICTURE VALUE
           0D        12  142A  3286          BNEQ    5$                             ; NOT A MARKER
           51        D7  142C  3287          DECL    R1                             ; CHECK FOR END
           F5        18  142E  3288          BGEQ    3$                             ; NOT YET
           51        D4  1430  3289          CLRL    R1
        30 A4        51  B0  1432  3290          MOVW    R1,TTY$W_RB_LINOFF(R4)     ; SAVE CURRENT OFFSET
           0076      31  1436  3291          BRW     FIELD_FULL                     ; FULL OF MARKERS
                     1439  3292 5$:
           50        93  1439  3293          BITB    R0,VERIFY_ARRAY[R3]            ; IS THIS CHARACTER VALID HERE
00000000'EF43                1438
           03        12  1441  3294          BNEQ    10$                            ; OK
           009C      31  1443  3295          BRW     RIGHT_TERM
                     1446  3296 ;
                     1446  3297 ; SHIFT DATA IN INPUT FIELD
                     1446  3298 ;
                     1446  3299 10$:
        30 A4        51  B0  1446  3300          MOVW    R1,TTY$W_RB_LINOFF(R4)     ; SAVE CURRENT OFFSET
                     144A  3301
           51        D4  144A  3302          CLRL    R1                             ; INIT DESTINATION POINTER
                     144C  3303
                     144C  3304 ; FIND FIRST VALID DESTINATION
                     144C  3305
                     144C  3306 20$:
     18 B441         95  144C  3307          TSTB    @TTY$L_RB_PIC(R4)[R1]          ; MARKER?
           08        12  1450  3308          BNEQ    30$                            ; NO, POSITION IS OK
           51        D6  1452  3309          INCL    R1                             ; LOOK FOR NEW DESTINATION
  3C A4    51        B1  1454  3310          CMPW    R1,TTY$W_RB_TXTOFF(R4)         ; PAST END?
           F2        1F  1458  3311          BLSSU   20$                            ; NO, IT IS STILL OK
        0052         31  145A  3312          BRW     FIELD_FULL
                     145D  3313 30$:
                     145D  3314 ;
                     145D  3315 ; IF NONCLR CHAR IN LEFTMOST DATA POSITION, FIELD IS ALREADY FULL, MUST
                     145D  3316 ;       BE NON-AUTOTAB. TERMINATE WITH OVERFLOWING CHAR
                     145D  3317 ;
     00 B441         91  145D  3318          CMPB    @TTY$L_RB_TXT(R4)[R1],TTY$B_RB_RVFFIL(R4);THIS A CLR CHAR?
        47 A4            1461
           07        13  1463  3319          BEQL    35$                            ;YES - NOT FULL YET
        3C A4        B0  1465  3320          MOVW    TTY$W_RB_TXTOFF(R4),-          ;SHOW OVERFLOW OFFSET
        30 A4            1468  3321                  TTY$W_RB_LINOFF(R4)            ;IF FIELD FULL
           76        11  146A  3322          BRB     RIGHT_TERM                     ; AND QUIT
                     146C  3323 35$:
           51        DD  146C  3324          PUSHL   R1                             ;SAVE OFFSET TO LEFT BYTE
                     146E  3325
     50    51        D0  146E  3326          MOVL    R1,R0                          ; INIT SOURCE POINTER
                     1471  3327
                     1471  3328 ; FIND NEW SOURCE
                     1471  3329
                     1471  3330 40$:
           50        D6  1471  3331          INCL    R0
  30 A4    50        B1  1473  3332          CMPW    R0,TTY$W_RB_LINOFF(R4)         ; PAST END?
```

```
                     12   1A  1477  3333          BGTRU   100$                                    ; YES THEN SHIFT COMPLETE
                 18 B440   95  1479  3334          TSTB    @TTY$L_RB_PIC(R4)[R0]                   ; MARKER?
                      F2   13  147D  3335          BEQL    40$                                     ; YES, LOOK AGAIN
                 00 B440   90  147F  3336          MOVB    @TTY$L_RB_TXT(R4)[R0],@TTY$L_RB_TXT(R4)[R1]; SHIFT THE DATA
                 00 B441       1483
                 51   50   D0  1486  3337          MOVL    R0,R1                                   ; OLD SOURCE IS NEW DESTINATION
                      E6   11  1489  3338          BRB     40$
                           148B  3339
                           148B  3340 ; SHIFT COMPLETE
                           148B  3341
                           148B  3342 100$:
                           148B  3343 ;
                           148B  3344 ;       UPCASE IF REQUESTED.
                           148B  3345 ;
                      08   E1  148B  3346          BBC     #TRM$V_TM_CVTLOW,-                      ; SKIP IF UPCASE NOT
                   0A 20 A4       148D  3347                  TTY$L_RB_MOD(R4),105$                   ;       REQUESTED
            0000'CF43   03   E1  1490  3348          BBC     #TTY$V_CA_LOWER,W^TTY$A_TYPE[R3],105$; is this lower case?
                           03   1496
                   53   20   8A  1497  3349          BICB    #^X20,R3                                ; yes MAKE LOWERCASE UPPER
                           149A  3350 105$:
                           149A  3351
                 00 B441   53   90  149A  3352          MOVB    R3,@TTY$L_RB_TXT(R4)[R1]                ; STORE NEW CHARACTER INTO
                           149F  3353                                                              ; END POSITION
                      51 8ED0   149F  3354          POPL    R1                                      ; GET START POSITION
                   00 B441   91  14A2  3355          CMPB    @TTY$L_RB_TXT(R4)[R1],TTY$B_RB_RVFCLR(R4);START WITH CLR CHAR?
                     46 A4       14A6
                      10   13  14A8  3356          BEQL    RIGHT_ECHO                              ;YES - FLD NOT FULL
                      12   E1  14AA  3357          BBC     #TRM$V_TM_AUTO_TAB,-                    ;FLD FULL - NONAUTOTAB
                   0B 20 A4       14AC  3358                  TTY$L_RB_MOD(R4),RIGHT_ECHO             ; JUST ECHO
                                 14AF  3359                                                          ;FLD FULL - AUTOTAB
                                 14AF  3360                                                          ; FALL THROUGH
                                 14AF  3361
                                 14AF  3362
                                 14AF  3363 FIELD_FULL:
                                 14AF  3364          SET_STATE EOL
                                 14B3  3365 10$:
               51   58 A5   D0  14B3  3366          MOVL    UCB$L_IRP(R5),R1                        ; GET IRP ADDRESS
                    38 A1   D4  14B7  3367          CLRL    IRP$L_MEDIA(R1)                         ; SIGNAL NO TERMINATOR
                                 14BA  3368
                                 14BA  3369 RIGHT_ECHO:
                  00FC C5   B4  14BA  3370          CLRW    UCB$W_TT_CURSOR(R5)
                                 14BE  3371          IF_STATE NOECHO,GODONE                          ; NO ECHO THEN DON'T ECHO
                         50 8ED0   14C2  3372          POPL    R0
                      32 A4   B4  14C5  3373          CLRW    TTY$W_RB_LINREST(R4)
               44 A4     0A   9B  14C8  3374          MOVZBW  #TTY$K_ER_RVECHO,TTY$W_RB_MODE(R4); ECHO THE LINE
            14 A4     4A A4   9E  14CC  3375          MOVAB   TTY$A_RB_PRM(R4),TTY$L_RB_ECHSTR(R4); THE ADDRESS OF THE STRING
            30 A4     34 A4   81  14D1  3376          ADDB3   TTY$W_RB_PRMLEN(R4),TTY$W_RB_LINOFF(R4),-; GET THE LENGTH TO ECHO
                      0B A4       14D6  3377                  TTY$B_RB_ECHLEN(R4)                     ;...
                      0B A4   96  14D8  3378          INCB    TTY$B_RB_ECHLEN(R4)                     ; ACCOUNT FOR ZERO BASE
                                 14DB  3379          SET_STATE EDITREAD
                       F874   31  14DF  3380          BRW     EDITREAD
                                 14E2  3381
                                 14E2  3382 .ENABLE LSB
                                 14E2  3383
                                 14E2  3384 RIGHT_TERM:
                                 14E2  3385          IF_STATE NOFLTR,3$                              ; SKIP EDITING IF NOFLTR
                         03   91  14E6  3386          CMPB    #TTY$K_ET_DELEFT,TTY$A_CCLIST[R3]; IS THIS A DELETE CHARACTER
```

```
00000000'EF43      14E8
          4D 13    14EE  3387          BEQL    DELRIGHT_JUST                   ; YES THEN DELETE
                   14F0  3388 3$:       ;IF NOT_STATE ESCAPE,4$                 ; ESCAPE PROCESSING THEN
          04 91    14F4  3389          CMPB    #TTY$K_ET_ESCAPE,TTY$A_CCLIST[R3]; ESCAPE ?
00000000'EF43      14F6
          28 13    14FC  3390          BEQL    10$                             ; YES
7E A5  3C A4 B0    14FE  3391 4$:       MOVW    TTY$W_RB_TXTOFF(R4),UCB$W_BCNT(R5); FILL REQUIRED FIELD
                   1503  3392          SET_STATE <EOL>                         ; SET END OF LINE
                   1507  3393 5$:
  51  58 A5 D0     1507  3394          MOVL    UCB$L_IRP(R5),R1                ; GET IRP ADDRESS
38 A1    53 98     150B  3395          MOVZBW  R3,IRP$L_MEDIA(R1)              ; TERMINATOR
3A A1    01 90     150F  3396          MOVB    #1,IRP$L_MEDIA+2(R1)            ; TERM LENGTH
      30 A4 90     1513  3397          MOVB    TTY$W_RB_LINOFF(R4),-
      3B A1        1516  3398                  IRP$L_MEDIA+3(R1)               ; OFFSET TO INVALID CHAR
  51  3C A4 3C     1518  3399          MOVZWL  TTY$W_RB_TXTOFF(R4),R1          ; GET THE ADDRESS OF THE END
   00 B441 9E      151C  3400          MOVAB   @TTY$L_RB_TXT(R4)[R1],TTY$L_RB_PIC(R4);OF THE INITIAL STRING
     18 A4         1520
18 B4    53 90     1522  3401          MOVB    R3,@TTY$L_RB_PIC(R4)            ; AND STORE THE CHARACTER THERE
        FE2C 31    1526  3402 GODONE:   BRW     DONE                            ; THEN EXIT
                   1529  3403 ;
                   1529  3404 ; SIGNAL ESCAPE SEQUENCE ACTIVE
                   1529  3405 ;
                   1529  3406 10$:
                   1529  3407          SET_STATE ESC                           ; ANNOUNCE ESCAPE STATE
          51 DD    152E  3408          PUSHL   R1                      ; SAVE R1
        EF70 30    1530  3409          BSBW    ESCINIT                 ; INIT THE ESCAPE SEQUENCE RULES
0103 C5   51 90    1533  3410          MOVB    R1,UCB$B_TT_ESC(R5)     ;
          51 8ED0  1538  3411          POPL    R1                      ; RESTORE R1
                   153B  3412
          CA 11    153B  3413          BRB     5$                              ; SKIP SETTING END
                   153D  3414
                   153D  3415          .DISABLE LSB
                   153D  3416 ;
                   153D  3417 ; DELETE RIGHT JUSTIFYED
                   153D  3418 ;
                   153D  3419 DELRIGHT_JUST:
50   51    D0      153D  3420          MOVL    R1,R0                           ; GET THE CURRENT LOCATION
     50    D7      1540  3421 10$:     DECL    R0                              ; MOVE RIGHT 1 CHARACTER
     12    19      1542  3422          BLSS    20$                             ; ECHO THE LINE
18 B440    95      1544  3423          TSTB    @TTY$L_RB_PIC(R4)[R0]           ; IS THIS A MARKER?
     F6    13      1548  3424          BEQL    10$                             ; YES THEN SKIP IT
00 B440    90      154A  3425          MOVB    @TTY$L_RB_TXT(R4)[R0],-         ; MOVE THE CHARACTER
00 B441            154E
                   1551  3426                  @TTY$L_RB_TXT(R4)[R1]
51   50    D0      1551  3427          MOVL    R0,R1
     EA    11      1554  3428          BRB     10$
                   1556  3429
  47 A4    90      1556  3430 20$:     MOVB    TTY$B_RB_RVFFIL(R4),-           ; MOVE A FILL CHARACTER FROM THE LEFT
00 B441            1559  3431                  @TTY$L_RB_TXT(R4)[R1]
    FF5B   31      155C  3432          BRW     RIGHT_ECHO                      ; NOW ECHO THE CHARACTER
```

```
                     155F  3434          .SBTTL  READ VERIFY ESCAPE TERMINATOR ROUTINE
                     155F  3435 ;++
                     155F  3436 ; DESCRIPTION:
                     155F  3437 ;       CALLED ON ALL THE CHARACTERS FOLLOWING THE ESCAPE
                     155F  3438 ;
                     155F  3439 ; INPUTS:
                     155F  3440 ;
                     155F  3441 ;       R2 - UNIT STATE VECTOR
                     155F  3442 ;       R4 - ADDPESS OF THE READ BUFFER
                     155F  3443 ;       R5 - UCB ADDRESS
                     155F  3444 ;
                     155F  3445 ; IMPLICIT INPUTS:
                     155F  3446 ;
                     155F  3447 ;       IRP$L_MEDIA+2 - CURRENT OFFSET TO IN THE BUFFER (ACCESSED AS A BYTE)
                     155F  3448 ;       TTY$L_RB_PIC  - THE ADDRESS OF THE ESCAPE CHARACTER IN THE BUFFER
                     155F  3449 ;       TTY$W_RB_TXTSIZ - LENGTH OF THE USERS BUFFER
                     155F  3450 ;--
                     155F  3451 ESCAPE_TERM:
 50    58 A5    D0   155F  3452          MOVL    UCB$L_IRP(R5),R0                ; GET IRP ADDRESS
 51    3A A0    9A   1563  3453          MOVZBL  IRP$L_MEDIA+2(R0),R1            ; GET CURRENT OFFSET
                     1567  3454
 40 A4    51    B1   1567  3455          CMPW    R1,TTY$W_RB_TXTSIZ(R4)          ; DO WE HAVE ROOM FOR THIS
                 14    1E   156B  3456          BGEQU   30$                     ;NO THEN TELL THEM ABOUT IT
18 B441    53    90   156D  3457          MOVB    R3,@TTY$L_RB_PIC(R4)[R1]        ; STORE THE DATA
       3A A0    96   1572  3458          INCB    IRP$L_MEDIA+2(R0)               ; BUMP TERM LENGTH
                     1575  3459
         EF3C    30   1575  3460          BSBW    ESCSYNTAX                       ; PARSE ESCAPE SEQUENCE
            0B   14   1578  3461          BGTR    10$                             ; SEQUENCE OK, CONTINUE
            0C   19   157A  3462          BLSS    20$                             ; SEQUENCE ERROR
                     157C  3463
                     157C  3464 ; SEQUENCE COMPLETED NORMALLY
                     157C  3465          CLR_STATE ESC                           ; CLEAR THE ESCAPE STATE THEN
                     1581  3466
                     1581  3467 30$:     SET_STATE EOL                           ; SET END OF LINE
                     1585  3468 10$:
FDCD    31           1585  3469          BRW     DONE
                     1588  3470
                     1588  3471 20$:
                     1588  3472          SET_STATE BADESC                        ; ESCAPE SEQ ERROR
  F3    11           158C  3473          BRB     30$                             ; TELL THE WORLD
```

```
158E  3475          .SBTTL  End of module
158E  3476 TT_END:
158E  3477          .END
```

ADJUST_CURSOR	000001AE	R	02	FORWARD_CHAR		00000C5C	R	02			
AESECHO	00000E00	R	02	FULL_1		0000135B	R	02			
BACKSPACING	000003FB	R	02	GETNEXTCHAR		00000005	R	02			
BACK_CHAR	00000896	R	02	GODEL		00000B5F	R	02			
BACK_TAB	00000EFA	R	02	GODONE		00001526	R	02			
BSPACE	0000020B	R	02	GODROP		00000C04	R	02			
BUFEMPTY	00000612	R	02	GOOUT		00000C 8	R	02			
BURST	00000099	R	02	INDELST		00000746	R	02			
CALLBACK	00000DE7	R	02	INI$BRK		********	X	02			
CARRIAGE	00000212	R	02	INSERT_CHAR		00000578	R	02			
CHAR_SIZE	00000F5D	R	02	INTEXIT		0000003D	R	02			
CHECK_BUFFER	00000059A	R	02	INTEXIT1		00000048	R	02			
CLRECHO	00000E0F	R	02	IO$M_TRMNOECHO	=	00001000					
CLRREST	00000E22	R	02	IO$V_CVTLOW	=	00000008					
CNT	= 00000001			IO$V_NOECHO	=	00000006					
COMECHO	00000E17	R	02	IO$V_TRMNOECHO	=	0000000C					
CROUTPUT	00000216	R	02	IRP$C_MEDIA	=	00000038					
CTRLR	000008D3	R	02	IRP$W_FUNC	=	00000020					
CTRLU	00000913	R	02	LEFT_JUSTIFY		000012C3	R	02			
CTRLZ	000002A3	R	02	LFOUTPUT		00000321	R	02			
CURSOROVRFLOW	0000040D	R	02	LINEFEED		0000031C	R	02			
CVTLOW	00000654	R	02	LOADADR		00001085	R	02			
DELCHAR	000009CE	R	02	MOVECURSOR		00000E55	R	02			
DELEFT_JUST	000013C9	R	02	MOVEREADATA		0000052D	R	02			
DELESCAPE	00000B4C	R	02	MOVE_BOL		00000C97	R	02			
DELETE_WORD	00000B65	R	02	MOVE_EOL		00000CE6	R	02			
DELRIGHT_JUST	0000153D	R	02	MOVIFNEC		00000DCA	R	02			
DISMISS	00000661	R	02	MULTIECHOING		00000465	R	02			
DONE	00001355	R	02	NORMAL		00000D77	R	02			
DROP	000002A0	R	02	NOTA		00000620	R	02			
DROP_CHAR	00000B62	R	02	OPEN		0000075D	R	02			
ECHLINE	00000E4F	R	02	OUTPUTANDWAIT		0000014F	R	02			
EDITINGCHAR	00000B0E	R	02	OUTPUTANDWAIT1		00000154	R	02			
EDITREAD	00000D56	R	02	OUTPUT_1		0000133D	R	02			
EOLSEEN	00000442	R	02	OUTPUT_S		00001328	R	02			
ESCAPE	000002CC	R	02	PASSALL		00001136	RG	02			
ESCAPE_CHAR	00000BFD	R	02	POWERREST		00000042	R	02			
ESCAPE_TAB	00000F9D	R	02	PRMECHO		00000E5C	R	02			
ESCAPE_TERM	0000155F	R	02	PRMECHO1		00000E79	R	02			
ESCINIT	000004A3	RG	02	QUOTE		000007F9	R	02			
ESCINPROG	00000666	R	02	QUOTING		00000D11	R	02			
ESCSYNTAX	000004B4	R	02	RDVERIFY		00001256	RG	02			
ESCSYNTAX_O	000004CA	R	02	RECALL		00000D19	R	02			
EXE$GL_ABSTIM	********	X	02	RECALLING		00000867	R	02			
EXIT	0000134B	R	02	RESTART		********	X	02			
EXITING	00000DD2	R	02	RIGHT_ECHO		000014BA	R	02			
E_SYNTAX	000004D7	RG	02	RIGHT_JUSTIFY		0000141C	R	02			
F	= 00000000			RIGHT_TERM		000014E2	R	02			
FIELD_FULL	000014AF	R	02	RVECHO		00000EA2	R	02			
FILLING	00000457	R	02	SENDLINEFEED		0000049A	R	02			
FIND_BOL	00000FCC	R	02	SIMCEOL		00000EC4	R	02			
FIND_BOL_NOCLEAR	00000FD0	RG	02	SS$_CONTROLO	=	00000609					
FORM	000003CD	R	02	SS$_DATAOVERUN	=	00000838					
FORMAT	000000DD	R	02	SS$_NORMAL	=	00000001					
FORMAT_CHAP	0000018E	R	02	SS$_TIMEOUT	=	0000022C					
FORMAT_LOCAL	000001EE	R	02	STOP_NOW		000000D6	R	02			
FORMAT_X	00000143	R	02	STRTMULTI		0000106B	RG	02			

```
STRTMULTI_1                     00001079 RG     02
STR_EXIT                        0000009E R      02
STR_TIMESET                     000000AF R      02
T                               = 00000028
TAB                             0000035F R      02
TABRIGHT                        00001098 R      02
TAB_BUFEMPTY                    0000050E R      02
TAB_CVTLOW                      00000527 R      02
TAB_DISMISS                     0000050B R      02
TAB_EDITINGCHAR                 0000051D R      02
TAB_EOLSEEN                     00000501 R      02
TAB_ESCINPROG                   00000520 R      02
TAB_GETNEXT                     00000517 R      02
TAB_INDELST                     00000511 R      02
TAB_LOCAL                       000003AC R      02
TAB_NOTA                        000004FE R      02
TAB_OPEN                        0000052A R      02
TAB_PASSALL                     00000514 R      02
TAB_QUOTE                       0000051A R      02
TERM                            00001371 R      02
TERMFOUND                       000005C3 R      02
TOGGELINSOV                     00000D49 R      02
TRM$V_TM_AUTO_TAB               = 00000012
TRM$V_TM_CVTLOW                 = 00000008
TRM$V_TM_NOECHO                 = 00000006
TRM$V_TM_NORECALL               = 00000010
TRM$V_TM_R_JUST                 = 00000011
TRM$V_TM_TRMNOECHO              = 0000000C
TT$V_LOWER                      = 00000007
TT$V_MECHFORM                   = 00000013
TT$V_MECHTAB                    = 00000008
TT$V_SCOPE                      = 0000000C
TT$V_WRAP                       = 00000009
TT$_VT100                       = 00000060
TT$_VT5X                        = 00000040
TT2$V_ANSICRT                   = 00000018
TT2$V_DECCRT                    = 0000001D
TT2$V_EDITING                   = 0000000C
TT2$V_LOCALECHO                 = 00000000
TTY$A_ANSIBACKUP                ********     X  02
TTY$A_ANSICEL                   ********     X  02
TTY$A_ANSI_DEOL                 ********     X  02
TTY$A_ANSI_UPCEL                ********     X  02
TTY$A_CCLIST                    ********     X  02
TTY$A_CTRLR                     ********     X  02
TTY$A_CTRLU                     ********     X  02
TTY$A_DELCRTTAB                 ********     X  02
TTY$A_ESCAPE                    ********     X  02
TTY$A_ESCINIT                   ********     X  02
TTY$A_EXITECHO                  ********     X  02
TTY$A_FCNTKN                    ********     X  02
TTY$A_FORM                      ********     X  02
TTY$A_LONGFORM                  ********     X  02
TTY$A_MAXTIME                   ********     X  02
TTY$A_PREFIX                    ********     X  02
TTY$A_RB_PRM                    = 0000004A
TTY$A_SPACEBACK                 ********     X  02
TTY$A_TAB                       ********     X  02
TTY$A_TA_RCL                    = 00000018
TTY$A_TYPE                      ********     X  02
TTY$A_VTAB                      ********     X  02
TTY$A_WORDTERM                  ********     X  02
TTY$B_RB_ECHLEN                 = 00000008
TTY$B_RB_RVFCLR                 = 00000046
TTY$B_RB_RVFFIL                 = 00000047
TTY$C_BLANK                     = 00000020
TTY$C_BS                        = 00000008
TTY$C_CR                        = 0000000D
TTY$C_CSI                       = 0000009B
TTY$C_CTRLB                     = 00000002
TTY$C_CTRLZ                     = 0000001A
TTY$C_DELETE                    = 0000007F
TTY$C_DOLLAR                    = 00000024
TTY$C_ESCAPE                    = 0000001B
TTY$C_LF                        = 0000000A
TTY$C_LOWESC1                   = 0000007D
TTY$C_LOWESC2                   = 0000007E
TTY$C_TAB                       = 00000009
TTY$GETNEXTCHAR                 00000000 RG     02
TTY$K_EDITNORMAL                = 00000004
TTY$K_ER_AESECHO                = 00000009
TTY$K_ER_CLRECHO                = 00000001
TTY$K_ER_CLRREST                = 00000006
TTY$K_ER_ECHLINE                = 00000002
TTY$K_ER_EXITING                = 00000004
TTY$K_ER_MOVECURSOR             = 00000005
TTY$K_ER_PRMECHO                = 00000007
TTY$K_ER_PRMECHO1               = 00000008
TTY$K_ER_RVECHO                 = 0000000A
TTY$K_ER_SIMCEOL                = 0000000B
TTY$K_ET_BACK_CHAR              = 00000005
TTY$K_ET_DELEFT                 = 00000003
TTY$K_ET_ESCAPE                 = 00000004
TTY$K_ET_FORWARD_CHAR           = 00000006
TTY$K_ET_RECALL                 = 0000000B
TTY$K_ET_TERMINATE              = 0000000F
TTY$K_ET_UNUSED                 = 0000000D
TTY$K_MAXESCTKN                 ********     X  02
TTY$L_RB_AES                    = 00000024
TTY$L_RB_DATA                   = 0000004A
TTY$L_RB_ECHSTR                 = 00000014
TTY$L_RB_LIN                    = 0000002C
TTY$L_RB_MOD                    = 00000020
TTY$L_RB_PIC                    = 00000018
TTY$L_RB_TERM                   = 0000001C
TTY$L_RB_TXT                    = 00000000
TTY$L_TA_DATA                   = 00000118
TTY$L_TA_END                    = 00000010
TTY$L_TA_GET                    = 00000004
TTY$L_WB_END                    = 00000020
TTY$L_WB_IRP                    = 00000024
TTY$L_WB_NEXT                   = 0000001C
TTY$M_CH_CTRL                   = 00000020
TTY$M_CH_CTRL2                  = 00000080
```

```
TTY$M_CH_CTRL3                  = 00000040
TTY$M_CH_SPEC                   = 00000010
TTY$M_ST_BACKSPACE              = 00000020
TTY$M_ST_BADESC                 = 00000100
TTY$M_ST_CTRLO                  = 00000001
TTY$M_ST_CTRLR                  = 00040000
TTY$M_ST_CURSOR                 = 00000008
TTY$M_ST_DEL                    = 00000002
TTY$M_ST_EDITING                = 00100000
TTY$M_ST_EDITREAD               = 00000200
TTY$M_ST_EOL                    = 00000100
TTY$M_ST_ESC                    = 00000080
TTY$M_ST_ESCAPE                 = 00000800
TTY$M_ST_ESC_O                  = 00004000
TTY$M_ST_FILL                   = 00000004
TTY$M_ST_MULTI                  = 00000040
TTY$M_ST_NINTMULTI              = 08000000
TTY$M_ST_NL                     = 00000200
TTY$M_ST_NOECHO                 = 00000008
TTY$M_ST_NOFLTR                 = 00000040
TTY$M_ST_OVERSTRIKE             = 00800000
TTY$M_ST_OVRFLO                 = 00010000
TTY$M_ST_PASALL                 = 00000004
TTY$M_ST_PRE                    = 04000000
TTY$M_ST_QUOTING                = 00400000
TTY$M_ST_RDVERIFY               = 00000400
TTY$M_ST_READ                   = 00001000
TTY$M_ST_RECALL                 = 00000800
TTY$M_ST_SENDLF                 = 00000010
TTY$M_ST_SKIPCRLF               = 00080000
TTY$M_ST_SKIPLF                 = 00002000
TTY$M_ST_TABEXPAND              = 00200000
TTY$M_ST_TABRIGHT               = 40000000
TTY$M_ST_TYPFUL                 = 00001000
TTY$M_ST_WRAP                   = 00008000
TTY$M_ST_WRITE                  = 00000080
TTY$M_ST_WRTALL                 = 00000010
TTY$Q_RB_ECHOAREA               = 0000000C
TTY$READONE                     ******** X   02
TTY$V_CH_LOWER                  = 00000003
TTY$V_CH_SPEC                   = 00000004
TTY$V_PC_NOCRLF                 = 00000007
TTY$V_PC_NOTIME                 = 00000000
TTY$V_RS_WRAP                   = 00000000
TTY$V_ST_DEL                    = 00000001
TTY$V_ST_ECHAES                 = 00000019
TTY$V_ST_EDITING                = 00000014
TTY$V_ST_ESC_O                  = 0000000E
TTY$V_ST_OVERSTRIKE             = 00000017
TTY$V_ST_PRE                    = 0000001A
TTY$V_ST_READ                   = 0000000C
TTY$V_ST_SKIPLF                 = 0000000D
TTY$V_SX_BACKSPACE              = 00000005
TTY$V_SX_BADESC                 = 00000028
TTY$V_SX_CTRLO                  = 00000020
TTY$V_SX_CTRLR                  = 00000032
TTY$V_SX_CURSOR                 = 00000003
TTY$V_SX_DEL                    = 00000021
TTY$V_SX_EDITING                = 00000034
TTY$V_SX_EDITREAD               = 00000009
TTY$V_SX_EOL                    = 00000008
TTY$V_SX_ESC                    = 00000027
TTY$V_SX_ESCAPE                 = 0000002B
TTY$V_SX_ESC_O                  = 0000002E
TTY$V_SX_FILL                   = 00000002
TTY$V_SX_MULTI                  = 00000006
TTY$V_SX_NINTMULTI              = 0000003B
TTY$V_SX_NL                     = 00000029
TTY$V_SX_NOECHO                 = 00000023
TTY$V_SX_NOFLTR                 = 00000026
TTY$V_SX_OVERSTRIKE             = 00000037
TTY$V_SX_OVRFLO                 = 00000030
TTY$V_SX_PASALL                 = 00000022
TTY$V_SX_PRE                    = 0000003A
TTY$V_SX_QUOTING                = 00000036
TTY$V_SX_RDVERIFY               = 0000000A
TTY$V_SX_READ                   = 0000000C
TTY$V_SX_RECALL                 = 0000000B
TTY$V_SX_SENDLF                 = 00000004
TTY$V_SX_SKIPCRLF               = 00000033
TTY$V_SX_SKIPLF                 = 0000002D
TTY$V_SX_TABEXPAND              = 00000035
TTY$V_SX_TABRIGHT               = 0000003E
TTY$V_SX_TERMNORM               = 00000038
TTY$V_SX_TYPFUL                 = 0000002C
TTY$V_SX_WRAP                   = 0000002F
TTY$V_SX_WRITE                  = 00000007
TTY$V_SX_WRTALL                 = 00000024
TTY$WRITEDONE                   ******** X   02
TTY$W_RB_AESLEN                 = 00000028
TTY$W_RB_CPZCUR                 = 00000038
TTY$W_RB_CPZORG                 = 0000003A
TTY$W_RB_ESCTKN                 = 00000048
TTY$W_RB_LINOFF                 = 00000030
TTY$W_RB_LINREST                = 00000032
TTY$W_RB_MODE                   = 00000044
TTY$W_RB_PRMLEN                 = 00000034
TTY$W_RB_RDSTATE                = 0000002A
TTY$W_RB_TXTOFF                 = 0000003C
TTY$W_RB_TXTSIZ                 = 00000040
TTY$W_TA_INAHD                  = 0000000C
TTY$W_TA_RCLOFF                 = 0000000E
TTY$W_TA_RCLSIZ                 = 00000014
TTY$W_WB_BCNT                   = 0000002A
TTY$W_WB_STATUS                 = 00000028
TTY$XON                         ******** X   02
TT_END                          0000158E R   02
UCB$B_DEVTYPE                   = 00000041
UCB$B_TT_CRFILL                 = 000000F6
UCB$B_TT_ESC                    = 00000103
UCB$B_TT_ESC_O                  = 00000104
UCB$B_TT_FILL                   = 00000102
UCB$B_TT_LASTC                  = 000000FF
UCB$B_TT_LFFILL                 = 000000F7
```

```
UCB$B_TT_LINE                   = 000000FE
UCB$B_TT_OUTYPE                 = 0000010B
UCB$L_DEVDEPEND                 = 00000044
UCB$L_DEVDEPND2                 = 00000048
UCB$L_DUETIM                    = 0000006C
UCB$L_IRP                       = 00000058
UCB$L_SVAPTE                    = 00000078
UCB$L_TT_MULTI                  = 000000D8
UCB$L_TT_OUTADR                 = 0000011C
UCB$L_TT_RDUE                   = 000000B0
UCB$L_TT_SMLT                   = 000000E0
UCB$L_TT_TYPAHD                 = 000000E4
UCB$L_TT_WRTBUF                 = 000000D4
UCB$M_INT                       = 00000002
UCB$M_TIM                       = 00000001
UCB$Q_TT_STATE                  = 000000B8
UCB$V_INT                       = 00000001
UCB$V_TT_TIMO                   = 00000001
UCB$W_BCNT                      = 0000007E
UCB$W_BOFF                      = 0000007C
UCB$W_BUFQUO                    = 00000018
UCB$W_DEVBUFSIZ                 = 00000042
UCB$W_DEVSTS                    = 00000068
UCB$W_STS                       = 00000064
UCB$W_TT_BSPLEN                 = 00000100
UCB$W_TT_CURSOR                 = 000000FC
UCB$W_TT_MULTILEN               = 000000DC
UCB$W_TT_OUTLEN                 = 00000120
UCB$W_TT_PRTCTL                 = 00000122
UCB$W_TT_SMLTLEN                = 000000DE
UPDATE_CURSOR                     000010B4 R      02
UPDCURSOR                         00000EF1 R      02
VERIFY_ARRAY                      ******** X      02
VTAB                              000003C5 R      02
W0                              = 00000000
W1                              = 00000001
WRITE_END                         00000C93 R      02
WRITING                           0000004D R      02
X                               = 00000000
X0                              = 00000000
X1                              = 00000001
XON                               0000C039 R      02
Z0                              = 00000003
Z1                              = 00000001
ZERO_CRZORG                       00001123 R      02
```

Psect synopsis

```
PSECT name                      Allocation              PSECT No.  Attributes
----------                      ----------              ---------  ----------
.  ABS  .                       00000000  (     0.)     00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD  NOWRT NOVEC BYTE
$ABS$                           00000000  (     0.)     01 (  1.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
$$$115_DRIVER                   0000158E  ( 5518.)     02 (  2.)  NOPIC   USR   CON   REL   LCL NOSHR   EXE   RD    WRT NOVEC LONG
```

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	96	00:00:00.22	00:00:00.82
Command processing	99	00:00:00.58	00:00:01.58
Pass 1	959	00:00:52.67	00:01:23.10
Symbol table sort	5	00:00:04.54	00:00:05.49
Pass 2	769	00:00:13.33	00:00:23.03
Symbol table output	1	00:00:00.31	00:00:00.57
Psect synopsis output	0	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	1931	00:01:11.67	00:01:54.61

The working set limit was 3150 pages.
273600 bytes (535 pages) of virtual memory were used to buffer the intermediate code.
There were 150 pages of symbol table space allocated to hold 2535 non-local and 289 local symbols.
3485 source lines were read in Pass 1, producing 29 object records in Pass 2.
52 pages of virtual memory were used to define 49 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA18:[SYS.OBJ]LIB.MLB;1	20
_255DUA18:[SYSLIB]STARLET.MLB;3	9
TOTALS (all libraries)	29

2791 GETS were required to define 29 macros.

There were no errors, warnings or information messages.

```
MACRO/LIS=LIS$:TTYCHARO/OBJ=OBJ$:TTYCHARO MSRC$:TTYCHARO/UPDATE=(BUG$:TTYCHARO)+EXECML$/LIB
```

0448 AH-EF71A-SE
VAX/VMS V4.1 SRC LST MCRF UPD

CSP
LIS

TTYCHARI
LIS

TTDRVR

OPDRVWS
LIS

TTDRIVER
MAP

YCDRIVER
MAP

CSPQUORUM
LIS

TTYCHARO
LIS

0449 AH-EF71A-SE
VAX/VMS V4.1 SRC LST MCRF UPD

YCDRIVER
LIS

TTYSTRSTP
LIS

TTYSUB
LIS