


```

CCCCCCCC HH   HH   MM   MM   EEEEEEEEEE   IIIIII   NN   NN   PPPPPPP   UU   UU   TTTTTTTTTT
CCCCCCCC HH   HH   MM   MM   EEEEEEEEEE   IIIIII:  NN   NN   PPPPPPP   UU   UU   TTTTTTTTTT
CC        HH   HH   MMMM  MMMM  EE           II      NN   NN   PP           PP   UU   UU   TT
CC        HH   HH   MMMM  MMMM  EE           II      NN   NN   PP           PP   UU   UU   TT
CC        HH   HH   MM   MM   MM   EE           II      NNNN  NN   PP           PP   UU   UU   TT
CC        HH   HH   MM   MM   MM   EE           II      NNNN  NN   PP           PP   UU   UU   TT
CC        HHHHHHHHHH  MM   MM   EEEEEEEEE   II      NN   NN   NN   PPPPPPP   UU   UU   TT
CC        HHHHHHHHHH  MM   MM   EEEEEEEEE   II      NN   NN   NN   PPPPPPP   UU   UU   TT
CC        HH   HH   MM   MM   EE           II      NN   NN   NN   PP           UU   UU   TT
CC        HH   HH   MM   MM   EE           II      NN   NN   NN   PP           UU   UU   TT
CC        HH   HH   MM   MM   EE           II      NN   NN   NN   PP           UU   UU   TT
CC        HH   HH   MM   MM   EE           II      NN   NN   NN   PP           UU   UU   TT
CC        HH   HH   MM   MM   EE           II      NN   NN   NN   PP           UU   UU   TT
CCCCCCCC HH   HH   MM   MM   EEEEEEEEEE   IIIIII   NN   NN   PP           UU   UU   TT
CCCCCCCC HH   HH   MM   MM   EEEEEEEEEE   IIIIII   NN   NN   PP           UUUUUUUUUU   TT

```

```

LL        IIIIII   SSSSSSSS
LL        IIIIII   SSSSSSSS
LL        II      SS
LL        II      SS
LL        II      SS
LL        II      SS
LL        II      SSSSSS
LL        II      SSSSSS
LL        II      SS
LL        II      SS
LL        II      SS
LL        IIIIII   SSSSSSSS
LLLLLLLLLL IIIIII   SSSSSSSS

```

```

1 0001 0 %TITLE 'EDT$CHMEINPUT - read with echo if possible'
2 0002 0 MODULE EDT$CHMEINPUT ( ! Read with echo if possible
3 0003 0 IDENT = 'V04-000' ! File: CHMEINPUT.BLI Edit: JBS1040
4 0004 0 ) =
5 0005 1 BEGIN
6 0006 1
7 0007 1 *****
8 0008 1 *
9 0009 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY *
10 0010 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS. *
11 0011 1 * ALL RIGHTS RESERVED. *
12 0012 1 *
13 0013 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED *
14 0014 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE *
15 0015 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER *
16 0016 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY *
17 0017 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY *
18 0018 1 * TRANSFERRED. *
19 0019 1 *
20 0020 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE *
21 0021 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT *
22 0022 1 * CORPORATION. *
23 0023 1 *
24 0024 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS *
25 0025 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL. *
26 0026 1 *
27 0027 1 *
28 0028 1 *****
29 0029 1
30 0030 1
31 0031 1 **
32 0032 1 FACILITY: EDT -- The DEC Standard Editor
33 0033 1
34 0034 1 ABSTRACT:
35 0035 1
36 0036 1 This module determines whether a special read can be performed
37 0037 1 which leaves character echoing to the terminal driver, and does the
38 0038 1 read if so.
39 0039 1
40 0040 1 ENVIRONMENT: Runs at any access mode - AST reentrant
41 0041 1
42 0042 1 AUTHOR: Bob Kushlis, CREATION DATE: Unknown
43 0043 1
44 0044 1 MODIFIED BY:
45 0045 1
46 0046 1 1-001 - Original. DJS 04-Feb-1981. This module was created by
47 0047 1 extracting the routine EDT$SRD ECHO from module CHANGE.BLI.
48 0048 1 1-002 - regularize headers. JBS 27-Feb-1981
49 0049 1 1-003 - Fix module name. JBS 02-Mar-1981
50 0050 1 1-004 - Revise journaling, and only do line reads if we can read at least
51 0051 1 four characters. JBS 18-Jun-1981
52 0052 1 1-005 - Prompt from the global string, if requested. JBS 21-Oct-1981
53 0053 1 1-006 - Remove length of prompt string. JBS 23-Oct-1981
54 0054 1 1-007 - Make the reads shorter to allow for the cursor positioning sequence.
55 0055 1 JBS 29-Jan-1982
56 0056 1 1-008 - Don't make the lengths of the reads depend on the special prompt;
57 0057 1 otherwise the QA system has trouble. JBS 29-Jan-1982

```

58 0058 1 1-009 - Add EDT\$SG JOU_VALID. JBS 09-Apr-1982
59 0059 1 1-010 - Worry about control C. JBS 24-May-1982
60 0060 1 1-011 - New screen update logic. JBS 13-Sep-1982
61 0061 1 1-012 - Include the EOL test routine, since it was only called from here. JBS 22-Sep-1982
62 0062 1 1-013 - Correct a misspelling in edit 1-012. JBS 23-Sep-1982
63 0063 1 1-014 - Add insert mode for VT102s. JBS 27-Sep-1982
64 0064 1 1-015 - Use a local text buffer, to avoid clobbering text if we are inserting. JBS 27-Sep-1982
65 0065 1 1-016 - Fix journaling of inserted text. JBS 28-Sep-1982
66 0066 1 1-017 - Remove EDT\$SG_LN_NO for new screen update logic. JBS 29-Sep-1982
67 0067 1 1-018 - Keep EDT\$SG_PRV_COL up to date. JBS 05-Oct-1982
68 0068 1 1-019 - Allow for fat characters to the right of the cursor. JBS 06-Oct-1982
69 0069 1 1-020 - Don't do optimized input if there is text on the message line. JBS 06-Oct-1982
70 0070 1 1-021 - Don't write out to the journal file here. STS 07-Oct-1982
71 0071 1 1-022 - Fix call to EDT\$SFMT_CHWID. JBS 13-Oct-1982
72 0072 1 1-023 - Don't send the CR and reposition the cursor unless the terminal
73 0073 1 driver needs it. JBS 16-Oct-1982
74 0074 1 1-024 - Handle some cases of DEL. JBS 10-Nov-1982
75 0075 1 1-025 - Don't redundantly enter insert mode. JBS 11-Nov-1982
76 0076 1 1-026 - Take into account characters already read when allowing for end of line. JBS 16-Nov-1982
77 0077 1 1-027 - Take into account characters already read when positioning the cursor. JBS 22-Nov-1982
78 0078 1 1-028 - Don't forget that DEL also repositions the cursor. JBS 24-Nov-1982
79 0079 1 1-029 - Don't forget to journal the DEL. Also, repaint the line if NOTRUNCATE. JBS 25-Nov-1982
80 0080 1 1-030 - Journal the correct text after DEL. JBS 25-Nov-1982
81 0081 1 1-031 - Change the call to EDT\$STST_KEYDEF. JBS 14-Dec-1982
82 0082 1 1-032 - Remove EDT\$SG_SHF. JBS 14-Dec-1982
83 0083 1 1-033 - Don't do it at the front of any line, even a continuation line. JBS 20-Dec-1982
84 0084 1 1-034 - Change the call to EDT\$SMRK_LNCHG. JBS 27-Dec-1982
85 0085 1 1-035 - Maintain EDT\$SG_CS_OLDCHNO. JBS 27-Dec-1982
86 0086 1 1-036 - If the screen is shifted don't do it. JBS 29-Dec-1982
87 0087 1 1-037 - Start on improving quality by going closer to the right margin before quitting. JBS 14-Jan-1982
88 0088 1 1-038 - Never read more than 70 characters at a time. JBS 08-Feb-1983
89 0089 1 1-039 - Read closer to the right margin. JBS 09-Feb-1983
90 0090 1 1-040 - Use EDT\$SC_INS_MODE to avoid entering and leaving insert mode so often. JBS 01-Apr-1983
91 0091 1 --
92 0092 1

```

: 94      0093 1 %SBTTL 'Declarations'
: 95      0094 1
: 96      0095 1 : TABLE OF CONTENTS:
: 97      0096 1 :
: 98      0097 1
: 99      0098 1 REQUIRE 'EDTSRC:TRANOJNAM';
100     0537 1
101     0538 1 FORWARD ROUTINE
102     0539 1     EDT$SRD_ECHO;
103     0540 1
104     0541 1 :
105     0542 1 : INCLUDE FILES:
106     0543 1 :
107     0544 1
108     0545 1 REQUIRE 'EDTSRC:EDTREQ';
109     0630 1
110     0681 1 :
111     0682 1 : MACROS:
112     0683 1 :
113     0684 1 :     NONE
114     0685 1 :
115     0686 1 : EQUATED SYMBOLS:
116     0687 1 :
117     0688 1 :
118     0689 1 LITERAL
119     0690 1     CHAR_LIMIT_1 = 2,
120     0691 1     CHAR_LIMIT_2 = 1;
121     0692 1
122     0693 1 :
123     0694 1 : OWN STORAGE:
124     0695 1 :
125     0696 1 :     NONE
126     0697 1 :
127     0698 1 : EXTERNAL REFERENCES:
128     0699 1 :
129     0700 1 :     In the routine
```

! Try to optimize terminal input

! Do optimized input even if only this many characters can be read
! Read this many chars less than we can

```
131 0701 1 %SBTTL 'EDT$$RD_ECHO - read with echo if possible'
132 0702 1
133 0703 1 GLOBAL ROUTINE EDT$$RD_ECHO ! Read with echo if possible
134 0704 1 =
135 0705 1
136 0706 1
137 0707 1 **
138 0708 1 FUNCTIONAL DESCRIPTION:
139 0709 1 This routine determines whether or not an optimization for terminal
140 0710 1 input can be done. If we are currently positioned at the end of a line,
141 0711 1 or if the terminal has the VT102 "insert character mode" feature,
142 0712 1 then it is possible to let the terminal driver do the echoing of printable
143 0713 1 characters for us up to the input of a character which may be a definable
144 0714 1 key, or to near the end of the line, where even a non-definable key needs
145 0715 1 special handling, such as wrap or display of a diamond. This is much more
146 0716 1 efficient than the single character input with no echo which is normally done.
147 0717 1
148 0718 1 This routine checks a whole series of conditions which must be met before
149 0719 1 optimized input is possible, then comes up with the number of
150 0720 1 characters which can be read with echo. If this is large enough then a special
151 0721 1 read routine is called to do the input. If the input is terminated by an
152 0722 1 escape or control character, that character will be put in the type-ahead
153 0723 1 character, so it will be the next character returned by EDT$$TI_INPCH.
154 0724 1
155 0725 1 FORMAL PARAMETERS:
156 0726 1
157 0727 1 NONE
158 0728 1
159 0729 1 IMPLICIT INPUTS:
160 0730 1
161 0731 1 EDT$$G_CUR_COL
162 0732 1 EDT$$G_CS [NO
163 0733 1 EDT$$A_SEC_BUF
164 0734 1 EDT$$G_KPAB
165 0735 1 EDT$$G_RCOV_MOD
166 0736 1 EDT$$A_CUR_BUF
167 0737 1 EDT$$G_TI_QID
168 0738 1 EDT$$G_WD_WRAP
169 0739 1 EDT$$T_LN_BUF
170 0740 1 EDT$$A_LN_PTR
171 0741 1 EDT$$A_LN_END
172 0742 1 EDT$$A_WK_LN
173 0743 1 EDT$$G_PRV_COL
174 0744 1 EDT$$T_PMT_KPD
175 0745 1 EDT$$G_TI_EDIT
176 0746 1 EDT$$G_MSGFLG
177 0747 1 EDT$$G_TI_DUMB
178 0748 1 EDT$$G_RDAHED
179 0749 1 EDT$$T_RDAHED
180 0750 1 EDT$$G_TRUN
181 0751 1 EDT$$A_CSR_SCRPTR
182 0752 1 EDT$$G_SHF
183 0753 1 EDT$$G_INSERT_MODE
184 0754 1
185 0755 1 IMPLICIT OUTPUTS:
186 0756 1
187 0757 1 EDT$$A_LN_PTR
```

```

188 0758 1 | EDTSSG_PRV COL
189 0759 1 | EDTSSG_LN_CHGD
190 0760 1 | EDTSSG_JOU_VALID
191 0761 1 | EDTSSG_CC_DONE
192 0762 1 | EDTSSG_RD_AHED
193 0763 1 | EDTSSG_VERT
194 0764 1 | EDTSSI_DEL_CH
195 0765 1 | EDTSSG_DEL_CHLEN
196 0766 1 | EDTSSG_CS_OLDCMNO
197 0767 1 |
198 0768 1 | ROUTINE VALUE:
199 0769 1 |
200 0770 1 |     0 = read with echo not possible, 1 = read with echo done.
201 0771 1 |
202 0772 1 | SIDE EFFECTS:
203 0773 1 |
204 0774 1 |     NONE
205 0775 1 |
206 0776 1 | --
207 0777 1 |
208 0778 2 | BEGIN
209 0779 2 |
210 0780 2 | EXTERNAL ROUTINE
211 0781 2 | EDTSSCHK_CC, | Test for a control C
212 0782 2 | EDTSSFMT_LIT : NOVALUE, | Format a literal string
213 0783 2 | EDTSSI_RDSTR : NOVALUE, | Read with echo
214 0784 2 | EDTSSI_BUFCH : NOVALUE, | Put characters in the journal buffer
215 0785 2 | EDTSSUPD_LNLEN : NOVALUE, | Update the length of the current line
216 0786 2 | EDTSSC_POSCSIF : NOVALUE, | Position the cursor if necessary
217 0787 2 | EDTSSC_NONREVID : NOVALUE, | End reverse video
218 0788 2 | EDTSSC_REVID : NOVALUE, | Start reverse video
219 0789 2 | EDTSSSEC_RNGPOS, | Compare the select line with the current line
220 0790 2 | EDTSSFMT_CHWID, | Compute the width of a character
221 0791 2 | EDTSTST_KEYDEF, | Test a key for a given definition
222 0792 2 | EDTSSC_ERATOEOI : NOVALUE, | Erase to end of line
223 0793 2 | EDTSMRR_LNCHG : NOVALUE, | Mark a line as having changed
224 0794 2 | EDTSSC_INS_MODE : NOVALUE; | Enter insert mode
225 0795 2 |
226 0796 2 | EXTERNAL
227 0797 2 | EDTSSG_CUR_COL, | current column
228 0798 2 | EDTSSG_LN_CHGD, | Indicates current line has changed.
229 0799 2 | EDTSSG_CS_LNO, | cursor line.
230 0800 2 | EDTSSA_SEC_BUF, | Pointer to select buffer.
231 0801 2 | EDTSSG_VERT, | Last entity was VERT flag.
232 0802 2 | EDTSSG_KPAD, | Keypad activated?
233 0803 2 | EDTSSG_RCOV_MOD, | In recovery mode?
234 0804 2 | EDTSSA_CUR_BUF : REF TBCB_BLOCK, | The current buffer tbcB
235 0805 2 | EDTSSG_TI_WID, | Width of terminal line
236 0806 2 | EDTSSG_WD_WRAP, | Word wrap
237 0807 2 | EDTSSI_LN_BUF, | Current line buffer
238 0808 2 | EDTSSA_LN_PTR : REF VECTOR [, BYTE], | Current character pointer
239 0809 2 | EDTSSA_LN_END, | Pointer to end of current line
240 0810 2 | EDTSSG_PRV_COL, | Previous column number
241 0811 2 | EDTSSI_PMT_KPAD : VECTOR [, BYTE], | Counted ASCII string for keypad prompt
242 0812 2 | EDTSSG_JOU_VALID, | The journal record is valid
243 0813 2 | EDTSSG_CC_DONE, | Control C actually aborted something
244 0814 2 | EDTSSA_WK_LN, | The current work line

```

```
245 0815 2 EDT$$Z_EOB_LN, ! The special line that marks end of buffer
246 0816 2 EDT$$G_TI_EDIT, ! 1 = this terminal has Insert Character Mode
247 0817 2 EDT$$G_MSGFLG, ! 1 = there is text on the message line
248 0818 2 EDT$$G_TI_DUMB, ! 1 = terminal driver needs CR to avoid wrapping lines
249 0819 2 EDT$$G_RDAHED, ! Number of chars in read-ahead buffer
250 0820 2 EDT$$T_RDAHED, ! The read-ahead buffer
251 0821 2 EDT$$T_DEL_CH : VECTOR [2, BYTE], ! Deleted character buffer.
252 0822 2 EDT$$G_DEL_CHLEN, ! Length of deleted character buffer
253 0823 2 EDT$$G_TRUN, ! 0 = NOTRUNCATE mode
254 0824 2 EDT$$A_CSR_SCRPTR : REF SCREEN_LINE, ! Pointer to current screen info for current line
255 0825 2 EDT$$G_CS_OLDCHNO, ! Old character position on the line
256 0826 2 EDT$$G_SHF, ! Screen shift amount
257 0827 2 EDT$$G_INSERT_MODE; ! 1 = screen is in insert mode
258 0828
259 0829 2 LOCAL
260 0830 2 BUF_LEFT,
261 0831 2 NUM_CHARS,
262 0832 2 READ,
263 0833 2 NUM_READ,
264 0834 2 READ_DONE,
265 0835 2 TERMINATOR_PROCESSED,
266 0836 2 TEXT_BUF : VECTOR [132, BYTE];
267 0837
268 0838 2 !+
269 0839 2 ! We can only do this in keypad mode.
270 0840 2 !-
271 0841 2
272 0842 2 IF ( NOT .EDT$$G_KPAD) THEN RETURN (0);
273 0843 2
274 0844 2 !+
275 0845 2 ! If we are on a continuation line don't do it.
276 0846 2 !-
277 0847 2
278 0848 2 IF (.EDT$$A_CSR_SCRPTR EQLA 0) THEN RETURN (0);
279 0849 2
280 0850 2 IF (.EDT$$A_CSR_SCRPTR [SCR_CHR_FROM] NEQ 0) THEN RETURN (0);
281 0851 2
282 0852 2 !+
283 0853 2 ! If we are at the left margin don't do it.
284 0854 2 !-
285 0855 2
286 0856 2 IF (.EDT$$A_CSR_SCRPTR [SCR_CHR_FROM] EQL (.EDT$$A_LN_PTR - EDT$$T_LN_BUF)) THEN RETURN (0);
287 0857 2
288 0858 2 !+
289 0859 2 ! If in recovery mode don't do it.
290 0860 2 !-
291 0861 2
292 0862 2 IF .EDT$$G_RCOV_MOD THEN RETURN (0);
293 0863 2
294 0864 2 !+
295 0865 2 ! If at end of buffer don't do it.
296 0866 2 !-
297 0867 2
298 0868 2 IF (.EDT$$A_WK_LN EQLA EDT$$Z_EOB_LN) THEN RETURN (0);
299 0869 2
300 0870 2 !+
301 0871 2 ! If there is text on the message line don't do it, since we want
```



```

: 302 0872 2 | to erase the text at the next keystroke. After that keystroke
: 303 0873 2 | the message line will be erased and we will come back here to
: 304 0874 2 | check again for optimized input.
: 305 0875 2 | -
: 306 0876 2 |
: 307 0877 2 | IF (.EDT$SG_MSGFLG) THEN RETURN (0);
: 308 0878 2 |
: 309 0879 2 | +
: 310 0880 2 | If the screen is shifted don't do it.
: 311 0881 2 | -
: 312 0882 2 |
: 313 0883 2 | IF (.EDT$SG_SHF NEQ 0) THEN RETURN (0);
: 314 0884 2 |
: 315 0885 2 | +
: 316 0886 2 | If this terminal has editing features don't do it if there is
: 317 0887 2 | a tab to the right of the cursor. If this terminal does not
: 318 0888 2 | have editing features, don't do it if there is anything to the
: 319 0889 2 | right of the cursor.
: 320 0890 2 | -
: 321 0891 2 |
: 322 0892 2 | IF .EDT$SG_TI_EDIT
: 323 0893 2 | THEN
: 324 0894 2 | BEGIN
: 325 0895 2 |
: 326 0896 2 | IF ( NOT CH$FAIL (CH$FIND_CH (CH$DIFF (.EDT$SA_LN_END, .EDT$SA_LN_PTR), .EDT$SA_LN_PTR, ASC_K_TAB)))
: 327 0897 2 | THEN
: 328 0898 2 | RETURN (0);
: 329 0899 2 |
: 330 0900 2 | END
: 331 0901 2 | ELSE
: 332 0902 2 |
: 333 0903 2 | IF (CH$DIFF (.EDT$SA_LN_END, .EDT$SA_LN_PTR) NEQ 0) THEN RETURN (0);
: 334 0904 2 |
: 335 0905 2 | +
: 336 0906 2 | Finally, it looks possible. Keep doing it as long as we can.
: 337 0907 2 | -
: 338 0908 2 | READ_DONE = 0;
: 339 0909 2 | READ = 0;
: 340 0910 2 |
: 341 0911 2 | DO
: 342 0912 2 | BEGIN
: 343 0913 2 | TERMINATOR_PROCESSED = 0;
: 344 0914 2 | +
: 345 0915 2 | Compute the number of characters left on the line.
: 346 0916 2 | -
: 347 0917 2 | NUM_CHARS = .EDT$SG_TI_WID - 1;
: 348 0918 2 | +
: 349 0919 2 | If we are in wrap mode, make sure we get control at the wrap column.
: 350 0920 2 | -
: 351 0921 2 |
: 352 0922 2 | IF (.EDT$SG_WD_WRAP LSS .NUM_CHARS) THEN NUM_CHARS = .EDT$SG_WD_WRAP;
: 353 0923 2 |
: 354 0924 2 | +
: 355 0925 2 | Subtract the current cursor position.
: 356 0926 2 | -
: 357 0927 2 | NUM_CHARS = .NUM_CHARS - .EDT$SG_CUR_COL - .READ;
: 358 0928 2 | +

```

```
359 0929 3 Subtract the width of the characters to the right of the cursor. Note that
360 0930 3 unless we are on a terminal with screen editing features this will always
361 0931 3 be zero. Note also that there can be no HTs to the right of the cursor,
362 0932 3 so the widths of the characters are independent of their position on the line.
363 0933 3 Hence the second parameter to EDT$FMT_CHWID will not be used.
364 0934 3 -
365 0935 3
366 0936 3 INCR CPTR FROM .EDT$A_LN_PTR TO .EDT$A_LN_END - 1 DO
367 0937 3 NUM_CHARS = .NUM_CHARS - EDT$FMT_CHWID (CH$RCHAR (.CPTR), 0);
368 0938 3
369 0939 3 +
370 0940 3 Make sure there is enough room left in the line buffer.
371 0941 3 -
372 0942 3 BUF_LEFT = 255 - CH$DIFF (.EDT$A_LN_PTR, EDT$T_LN_BUF);
373 0943 3
374 0944 3 IF (.BUF_LEFT LSS .NUM_CHARS) THEN NUM_CHARS = .BUF_LEFT;
375 0945 3
376 0946 3 +
377 0947 3 Don't try to read more than the space we have in our local buffer.
378 0948 3 -
379 0949 3
380 0950 3 IF ((.NUM_CHARS + .READ) GTR 132) THEN NUM_CHARS = 132 - .READ;
381 0951 3
382 0952 3 +
383 0953 3 Now, if we have a reasonable size, we can read with echo.
384 0954 3 -
385 0955 3
386 0956 3 IF (.NUM_CHARS GTR CHAR_LIMIT_1)
387 0957 3 THEN
388 0958 3 BEGIN
389 0959 3 +
390 0960 3 We will do a read with echo. Make sure the video attributes are right.
391 0961 3 -
392 0962 3
393 0963 3 IF (.EDT$A_SEL_BUF EQL .EDT$A_CUR_BUF)
394 0964 3 THEN
395 0965 3 BEGIN
396 0966 3
397 0967 3 IF (EDT$SEL_RNGPOS () LEQ 0) THEN EDT$SC_REVID () ELSE EDT$SC_NONREVID ()
398 0968 3
399 0969 3 END
400 0970 3 ELSE
401 0971 3 EDT$SC_NONREVID ();
402 0972 3
403 0973 3 +
404 0974 3 If we are not at the end of the line, put the terminal in insert mode. This can only
405 0975 3 be done on terminals that have the 'edit' feature.
406 0976 3 -
407 0977 3
408 0978 3 IF ((CH$DIFF (.EDT$A_LN_END, .EDT$A_LN_PTR) NEQ 0) AND (.EDT$G_INSERT_MODE EQL 0))
409 0979 3 THEN
410 0980 3 EDT$SC_INS_MODE ();
411 0981 3
412 0982 3 +
413 0983 3 Put out a carriage return to make the terminal driver think we are at the
414 0984 3 beginning of a line, then reposition the cursor. This is needed only for
415 0985 3 some terminal drivers, that lose track of the cursor and output a CRLF
```

```
416 0986 4 | if they think that the user is about to type to the right of the screen.
417 0987 4 | -
418 0988 4 |
419 0989 4 |     IF .EDT$$G_TI_DUMB
420 0990 4 |     THEN
421 0991 5 |         BEGIN
422 0992 5 |         EDT$$FMT_LIT (UPLIT (%STRING (%CHAR (ASC_K_CR))), 1);
423 0993 5 |         EDT$$G_PRV_COL = 0;
424 0994 4 |         END;
425 0995 4 |
426 0996 4 | +
427 0997 4 | | Make sure the cursor is positioned correctly.
428 0998 4 | | -
429 0999 4 | |     EDT$$SC_POSCSIF (.EDT$$G_CS_LNO, .EDT$$G_CUR_COL + .READ);
430 1000 4 | +
431 1001 4 | | Do the special read with echo.  Optionally prompt.  Since the terminal driver may
432 1002 4 | | count the length of the prompt, it must be short enough that our 'worst case' estimate
433 1003 4 | | of 10 characters in the repositioning allows for it.  Don't read more than 70 characters
434 1004 4 | | at a time.
435 1005 4 | | -
436 1006 4 | |
437 1007 4 | |     IF (.EDT$$T_PMT_KPD [0] GTR 0) THEN EDT$$FMT_LIT (EDT$$T_PMT_KPD [1], .EDT$$T_PMT_KPD [0]);
438 1008 4 | |
439 1009 4 | |     EDT$$TI_RDSTR (TEXT_BUF [.READ], MIN (70, .NUM_CHARS - CHAR_LIMIT_2), NUM_READ);
440 1010 4 | |     EDT$$G_PRV_COL = .EDT$$G_PRV_COL + .NUM_READ;
441 1011 4 | |     READ_DONE = 1;
442 1012 4 | |     END
443 1013 4 | | ELSE
444 1014 4 | |     NUM_READ = 0;
445 1015 4 | |
446 1016 4 | +
447 1017 4 | | Cause the characters to appear in the next journal record.
448 1018 4 | | -
449 1019 4 | |
450 1020 4 | |     INCR COUNTER FROM 0 TO (.NUM_READ - 1) DO
451 1021 4 | |     EDT$$TI_BUFCH (.TEXT_BUF [.READ + .COUNTER]);
452 1022 4 | |
453 1023 4 | |     EDT$$G_JOU_VALID = 1;
454 1024 4 | |     READ = .READ + .NUM_READ;
455 1025 4 | +
456 1026 4 | | If the line was terminated by a control C bail out.  If any characters were
457 1027 4 | | read the insert is aborted; otherwise the control C is effectively ignored.
458 1028 4 | | -
459 1029 4 | |
460 1030 4 | |     IF EDT$$CHK_CC ()
461 1031 4 | |     THEN
462 1032 4 | |     BEGIN
463 1033 4 | |
464 1034 4 | |     IF (.READ GTR 0) THEN EDT$$G_CC_DONE = 1;
465 1035 4 | |
466 1036 4 | |     RETURN (0);
467 1037 4 | |     END;
468 1038 4 | |
469 1039 4 | +
470 1040 4 | | If there is a single terminator, and if it is defined to delete
471 1041 4 | | the last character, shorten the string by one and do another read.
472 1042 4 | | -
```

```
473 1043 3
474 1044 4 IF ((.EDT$$G_RDAHED EQL 1) AND
475 1045 4 EDT$$ST_KEYDEF (CH$RCHAR (EDT$$T_RDAHED), UPLIT (BYTE ('D-C.')), 4, 0) AND
476 1046 4 (.READ GEQ 1))
477 1047 3 THEN
478 1048 4 BEGIN
479 1049 4 +
480 1050 4 | Make sure the delete character appears in the journal.
481 1051 4 -
482 1052 4 EDT$$TI_BUFCH (CH$RCHAR (EDT$$T_RDAHED));
483 1053 4 READ = .READ - 1;
484 1054 4 EDT$$SC_POSCSIF (.EDT$$G_CS_LNO, .EDT$$G_PRV_COL - 1);
485 1055 4
486 1056 5 IF (.EDT$$G_INSERT_MODE NEQ 0)
487 1057 4 THEN
488 1058 5 BEGIN
489 1059 5 +
490 1060 5 | We must delete exactly one character.
491 1061 5 -
492 1062 5 EDT$$FMT_LIT (UPLIT (BYTE (ASC_K_ESC, %C'E', %C'P')), 3);
493 1063 5 END
494 1064 4 ELSE
495 1065 5 BEGIN
496 1066 5 +
497 1067 5 | We are just before the character to delete, and there are no visible characters after
498 1068 5 | the character to delete. We can erase to end of line.
499 1069 5 -
500 1070 5 EDT$$SC_ERATOEOL ();
501 1071 4 END;
502 1072 4
503 1073 4 +
504 1074 4 | Store the character deleted in the delete character buffer.
505 1075 4 -
506 1076 4 EDT$$G_DEL_CHLEN = 1;
507 1077 4 EDT$$T_DEL_CH [0] = DIR_BACKWARD;
508 1078 4 EDT$$T_DEL_CH [1] = .TEXT_BUF [.READ];
509 1079 4 EDT$$G_RDAHED = 0;
510 1080 4 EDT$$G_VERT = 0;
511 1081 4 TERMINATOR_PROCESSED = 1;
512 1082 3 END;
513 1083 3
514 1084 3 +
515 1085 3 | Keep reading if we processed the terminator.
516 1086 3 -
517 1087 3 END
518 1088 2 UNTIL ( NOT .TERMINATOR_PROCESSED);
519 1089 2
520 1090 2 +
521 1091 2 | Insert the characters read into the line.
522 1092 2 -
523 P 1093 2 EDT$$CPY_MEM (CH$DIFF (.EDT$$A_LN_END, .EDT$$A_LN_PTR), .EDT$$A_LN_PTR,
524 1094 2 CH$P[US (.EDT$$A_LN_PTR, .READ));
525 1095 2 EDT$$CPY_MEM (.READ, TEXT_BUF [0], .EDT$$A_LN_PTR);
526 1096 2 +
527 1097 2 | Add the number of characters read to the line size.
528 1098 2 -
529 1099 2 EDT$$UPD_LNLEN (.READ);
```


				OFFC 00000	.EXTRN EDT\$\$G_RDAHED, EDT\$\$T_RDAHED	
					.EXTRN EDT\$\$T_DEL_CH, EDT\$\$G_DEL_CHLEN	
					.EXTRN EDT\$\$G_TRUN, EDT\$\$A_CSR_SCRPTR	
					.EXTRN EDT\$\$G_CS_OLDCHNO	
					.EXTRN EDT\$\$G_SHF, EDT\$\$G_INSERT_MODE	
					.ENTRY EDT\$\$RD_ECHO, Save R2,R3,R4,R5,R6,R7,R8,R9,-;	0703
					R10,R11	
				5B 00000000G 00 9E 00002	MOVAB EDT\$\$G_CUR_COL, R11	
				5A 00000000G 00 9E 00009	MOVAB EDT\$\$A_LN_END, R10	
				59 00000000G 00 9E 00010	MOVAB EDT\$\$A_LN_PTR, R9	
				5E FF78 CE 9E 00017	-136(SP), SP	
				03 00000000G 00 E8 0001C	BLBS EDT\$\$G_KPAD, 2\$	0842
					BRW 31\$	
				50 00000000G 00 D0 00026	MOVL EDT\$\$A_CSR_SCRPTR, R0	0848
					BEQL 1\$	
					TSTB 9(R0)	0850
					BNEQ 1\$	
				52 00000000G 69 D0 00034	MOVL EDT\$\$A_LN_PTR, R2	0856
				51 00000000G 00 9E 00037	MOVAB EDT\$\$T_LN_BUF, R1	
				52 00000000G 51 C3 0003E	SUBL3 R1, R2, RT	
51	09	51		08 00 00 00042	CMPZV #0, #8, 9(R0), R1	
					BEQL 1\$	
				D2 00000000G 00 E8 0004A	BLBS EDT\$\$G_RCOV_MOD, 1\$	0862
				50 00000000G 00 9E 00051	MOVAB EDT\$\$Z_EOB_CN, R0	0868
				50 00000000G 00 D1 00058	CML EDT\$\$A_WK_CN, R0	
					BEQL 1\$	
				BB 00000000G 00 E8 00061	BLBS EDT\$\$G_MSGFLG, 1\$	0877
				00000000G 00 D5 00068	TSTL EDT\$\$G_SHF	0883
					BNEQ 1\$	
				50 00000000G 6A D0 00070	MOVL EDT\$\$A_LN_END, R0	0896
				10 00000000G 00 E9 00073	BLBC EDT\$\$G_TI_EDIT, 4\$	0892
		51		50 00000000G 52 C3 0007A	SUBL3 R2, R0, RT	0896
		62		51 00000000G 09 3A 0007E	LOCC #9, R1, (R2)	
					BNEQ 3\$	
					CLRL R1	
					TSTL R1	
					BRB 5\$	
				52 00000000G 50 D1 0008A	CML R0, R2	0903
					BNEQ 1\$	
					READ	0909
					CLRL TERMINATOR_PROCESSED	0913
				52 00000000G 00 01 C3 00093	SUBL3 #1, EDT\$\$G_TI_WID, NUM_CHARS	0917
				50 00000000G 00 D0 0009B	MOVL EDT\$\$G_WD_WRAP, R0	0922
				52 00000000G 50 D1 000A2	CML R0, NUM_CHARS	
					BGEQ 7\$	
					MOVL R0, NUM_CHARS	
				50 00000000G 52 6B C3 000AA	SUBL3 EDT\$\$G_CUR_COL, NUM_CHARS, R0	0927
				52 00000000G 50 57 C3 000AE	SUBL3 READ, R0, NUM_CHARS	
					MOVL EDT\$\$A_LN_END, R5	0936
				54 00000000G 69 01 C3 000B5	SUBL3 #1, EDT\$\$A_LN_PTR, CPTR	
					BRB 9\$	
					CLRL -(SP)	0937
					MOVZBL (CPTR), -(SP)	
				00000000G 00 02 FB 000C0	CALLS #2, EDT\$\$FMT_CHWID	
					SUBL2 R0, NUM_CHARS	
				ED 00000000G 54 55 F2 000CA	AOBLSS R5, CPTR, 8\$	

50	0000C000G	00	9E	000CE	MOVAB	EDT\$\$T_LN_BUF, R0	0942		
50		69	C2	000D5	SUBL2	EDT\$\$A_LN_PTR, R0			
53	00FF	C0	9E	000D8	MOVAB	255(R0), BUF_LEFT			
52		53	D1	000DD	CMPL	BUF_LEFT, NUM_CHARS	0944		
		03	18	000E0	BGEQ	10\$			
52		53	D0	000E2	MOVL	BUF_LEFT, NUM_CHARS			
50	00000084	8F	57	C1	000E5	10\$: ADDL3	0950		
		50	D1	000E9	CMPL	READ, NUM_CHARS, R0			
		08	15	000F0	BLEQ	R0, #132			
52	00000084	8F	57	C3	000F2	SUBL3	0956		
		02	52	D1	000FA	11\$: CMPL	NUM_CHARS, #2		
		03	14	000FD	BGTR	12\$			
		00AD	31	000FF	BRW	19\$			
00000000G	00	00000000G	00	D1	00102	12\$: CMPL	EDT\$\$A_SEL_BUF, EDT\$\$A_CUR_BUF		
			14	12	0010D	BNEQ	13\$		
00000000G	00		00	FB	0010F	CALLS	#0, EDT\$\$SEL_RNGPOS		
			50	D5	00116	TSTL	R0		
			09	14	00118	BGTR	13\$		
00000000G	00		00	FB	0011A	CALLS	#0, EDT\$\$SC_REVID		
			07	11	00121	BRB	14\$		
00000000G	70		00	FB	00123	13\$: CALLS	#0, EDT\$\$SC_NONREVID		
	69		6A	D1	0012A	14\$: CMPL	EDT\$\$A_LN_END, EDT\$\$A_LN_PTR		
			0F	13	0012D	BEQL	15\$		
		00000000G	00	D5	0012F	TSTL	EDT\$\$G_INSERT_MODE		
			07	12	00135	BNEQ	15\$		
00000000G	00		00	FB	00137	CALLS	#0, EDT\$\$SC_INS_MODE		
		13	00000000G	00	E9	0013E	15\$: BLBC	EDT\$\$G_TI_DUMB, 16\$	
				01	DD	00145	PUSHL	#1	
		FEAA	CF	9F	00147	PUSHAB	P.AAA		
00000000G	00		02	FB	0014B	CALLS	#2, EDT\$\$FMT_LIT		
		00000000G	00	D4	00152	CLRL	EDT\$\$G_PRV_COL		
			50	D0	00158	16\$: MOVL	EDT\$\$G_CUR_COL, R0		
			6740	9F	0015B	PUSHAB	(READ)[R0]		
		00000000G	00	DD	0015E	PUSHL	EDT\$\$G_CS_LNO		
00000000G	00		02	FB	00164	CALLS	#2, EDT\$\$SC_POSCSIF		
		50	00000000G	00	9A	0016B	MOVZBL	EDT\$\$T_PMT_RPD, R0	
				0F	15	00172	BLEQ	17\$	
				50	DD	00174	PUSHL	R0	
		00000000G	00	9F	00176	PUSHAB	EDT\$\$T_PMT_KPD+1		
00000000G	00		02	FB	0017C	CALLS	#2, EDT\$\$FMT_LIT		
			5E	DD	00183	17\$: PUSHL	SP		
			50	FF	A2	9E	00185	MOVAB	-1(R2), R0
			50	DD	00189	PUSHL	R0		
00000046	8F		6E	D1	0018B	CMPL	(SP), #70		
			04	15	00192	BLEQ	18\$		
		6E	46	8F	9A	00194	MOVZBL	#70, (SP)	
		0C	AE47	9F	00198	18\$: PUSHAB	TEXT_BUF[READ]		
00000000G	00		03	FB	0019C	CALLS	#3, EDT\$\$TI_RDSTR		
00000000G	00		6E	C0	001A3	ADDL2	NUM_READ, EDT\$\$G_PRV_COL		
			58	01	D0	001AA	MOVL	#1, READ_DONE	
			02	11	001AD	BRB	20\$		
			6E	D4	001AF	19\$: CLRL	NUM_READ		
			54	01	CE	001B1	20\$: MNEGL	#1, COUNTER	
			10	11	001B4	BRB	22\$		
50	57		54	C1	001B6	21\$: ADDL3	COUNTER, READ, R0		
	7E	04	AE40	9A	001BA	MOVZBL	TEXT_BUF[R0], -(SP)		
00000000G	00		01	FB	001BF	CALLS	#1, EDT\$\$TI_BUFCH		

.....

EC	54	6E	F2	001C6	22\$:	AOBLSS	NUM_READ, COUNTER, 21\$	
	00000000G	00	01	D0 001CA		MOVL	#1, EDTSSG_JOU_VALID	1023
		57	6E	C0 001D1		ADDL2	NUM_READ, READ	1024
	00000000G	00	00	FB 001D4		CALLS	#0, EDTSSCHK_CC	1030
		0E	50	E9 001DB		BLBC	R0, 24\$	
			57	D5 001DE		TJIL	READ	1034
			07	15 001E0		BLEQ	23\$	
	00000000G	00	01	D0 001E2		MOVL	#1, EDTSSG_CC_DONE	
			00D2	31 001E9	23\$:	BRW	31\$	1036
		01	00G00000G	00 D1 001EC	24\$:	CMPL	EDTSSG_RDAHED, #1	1044
				03 13 001F3		BEQL	25\$	
			0084	31 001F5		BRW	28\$	
		7E		04 7D 001F8	25\$:	MOVQ	#4, -(SP)	1045
			F DFA	CF 9F 001FB		PUSHAB	P.AAB	
	00000000G	7E	00000000G	00 9A 001FF		MOVZBL	EDTSSST_RDAHED, -(SP)	
		00		04 FB 00206		CALLS	#4, EDTSSST_KEYDEF	
		6C		50 E9 0020D		BLBC	R0, 28\$	
				57 D5 00210		TSTL	READ	1046
				68 15 00212		BLEQ	28\$	
	00000000G	7E	00000000G	00 9A 00214		MOVZBL	EDTSSST_RDAHED, -(SP)	1052
		00		01 FB 0021B		CALLS	#1, EDTSSSTI_BUFCH	
				57 D7 00222		DECL	READ	1053
	7E	00000000G	00	01 C3 00224		SUBL3	#1, EDTSSG_PRV_COL, -(SP)	1054
			00000000G	00 DD 0022C		PUSHL	EDTSSG_CS_CNO	
	00000000G	00		02 FB 00232		CALLS	#2, EDTSSSC_POSCSIF	
			00000000G	00 D5 00239		TSTL	EDTSSG_INSERT_MODE	1056
				0F 13 0023F		BEQL	26\$	
				03 DD 00241		PUSHL	#3	1062
			F DB6	CF 9F 00243		PUSHAB	P.AAC	
	00000000G	00		02 FB 00247		CALLS	#2, EDTSSFMT_LIT	
				07 11 0024E		BRB	27\$	1056
	00000000G	00		00 FB 00250	26\$:	CALLS	#0, EDTSSSC_ERATOEOL	1070
	00000000G	00		01 D0 00257	27\$:	MOVL	#1, EDTSSG_DEL_CHLEN	1076
			00000000G	00 94 0025E		CLRB	EDTSSST_DEL_CH	1077
	00000000G	00	04 AE	47 90 00264		MOVB	TEXT_BUF[READ], EDTSSST_DEL_CH+1	1078
			00000000G	00 D4 0026D		CLRL	EDTSSG_RDAHED	1079
			00000000G	00 D4 00273		CLRL	EDTSSG_VERT	1080
		56		01 D0 00279		MOVL	#1, TERMINATOR_PROCESSED	1081
		03		56 E9 0027C	28\$:	BLBC	TERMINATOR_PROCESSED, 29\$	1088
			F E0F	31 0027F		BRW	6\$	
		56		69 D0 00282	29\$:	MOVL	EDTSSA_LN_PTR, R6	1094
	50	6A		56 C3 00285		SUBL3	R6, EDTSSA_LN_END, R0	
	6746	66		50 28 00289		MOVC3	R0, (R6), (READ)[R6]	
	66	04 AE		57 28 0028E		MOVC3	READ, TEXT_BUF, (R6)	1095
				57 DD 00293		PUSHL	READ	1099
	00000000G	00		01 FB 00295		CALLS	#1, EDTSSUPD_LNLEN	
				57 D5 0029C		TSTL	READ	1104
				10 13 0029E		BEQL	30\$	
		6B		57 C0 002A0		ADDL2	READ, EDTSSG_CUR_COL	1107
			00000000G	00 D4 002A3		CLRL	EDTSSG_VERT	1108
	00000000G	00		01 D0 002A9		MOVL	#1, EDTSSG_LN_CHGD	1115
		69		57 C0 002B0	30\$:	ADDL2	READ, EDTSSA_LN_PTR	1121
	00000000G	00		57 C0 002B3		ADDL2	READ, EDTSSG_CS_OLDCHNO	1122
		50		58 D0 002BA		MOVL	READ_DONE, R0	1123
				04 002BD		RET		
			50	D4 002BE	31\$:	CLRL	R0	1124
				04 002C0		RET		

EDT\$CHMEINPUT
V04-000

EDT\$CHMEINPUT - read with echo if possible
EDT\$SRD_ECHO - read with echo if possible

^{D 1}
15-Sep-1984 23:50:01
14-Sep-1984 12:22:26

VAX-11 Bliss-32 V4.0-742
[EDT.SRC]CHMEINPUT.BLI;1

Page 15
(3)

ED
VC

: Routine Size: 705 bytes, Routine Base: _EDT\$CODE + 000B

: 555 1125 1
: 556 1126 1 !<BLF/PAGE>



EDT\$CHMEINPUT
V04-000

EDT\$CHMEINPUT - read with echo if possible
EDT\$\$RD_ECHO - read with echo if possible

E 1
15-Sep-1984 23:50:01 VAX-11 Bliss-32 V4.0-742
14-Sep-1984 12:22:26 [EDT.SRC]CHMEINPUT.BLI;1

Page 16
(4)

: 558 1127 1 END
: 559 1128 1
: 560 1129 0 ELUDOM

! of module EDT\$CHMEINPUT

PSECT SUMMARY

Name Bytes Attributes
:_EDT\$CODE 716 NOVEC, NOWRT, RD, EYE, SHR, LCL, REL, CON, PIC, ALIGN(2)

Library Statistics

File	Total	Symbols Loaded	Percent	Pages Mapped	Processing Time
\$_\$255\$DUA28:[EDT.SRC]EDT.L32;1	377	39	10	40	00:00.2
\$_\$255\$DUA28:[EDT.SRC]PSECTS.L32;1	2	1	50	7	00:00.1

COMMAND QUALIFIERS

BLISS/CHECK=(FIELD, INITIAL, OPTIMIZE)/NOTRACEBACK/LIS=LISS:CHMEINPUT/OBJ=OBJ\$:CHMEINPUT MSRC\$:CHMEINPUT.BLI/UPDATE=(ENH\$:C
HMEINPUT)

: Size: 705 code + 11 data bytes
: Run Time: 00:30.2
: Elapsed Time: 00:34.5
: Lines/CPU Min: 2240
: Lexemes/CPU-Min: 7145
: Memory Used: 198 pages
: Compilation Complete

CHMFINENT LIS	CHMINIT LIS
CHMGOUNT LIS	CHMGINSTR LIS
CHMGSUSTR LIS	CHMINSMOD LIS
CHMEMESS LIS	CHMINSTAB LIS
CHMENTRM LIS	CHMINSCHR LIS
CHMEXVERB LIS	CHMINDATE LIS
CHMFINSTR LIS	CHMGDTR LIS
CHMGQSTR LIS	CHMLPKPD LIS
CHMINSSTR LIS	CHMKEYWRD LIS
CHMENDWRD LIS	CHMEXCOM LIS