

EEEEEEEEEE	DDDDDDDD	TTTTTTTTTT	
EEEEEEEEEE	DDDDDDDD	TTTTTTTTTT	
EE	DD DD	TT	
EE	DD DD	TT	
EE	DD DD	TT	
EE	DD DD	TT	
EEEEEEEEEE	DD DD	TT	
EEEEEEEEEE	DD DD	TT	
EE	DD DD	TT	
EE	DD DD	TT	
EE	DD DD	TT
EEEEEEEEEE	DDDDDDDD	TT
EEEEEEEEEE	DDDDDDDD	TT

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



0001 0
0002 0
0003 0
0004 0
0005 0
0006 0
0007 0
0008 0
0009 0
0010 0
0011 0
0012 0
0013 0
0014 0
0015 0
0016 0
0017 0
0018 0
0019 0
0020 0
0021 0
0022 0
0023 0
0024 0
0025 0
0026 0
0027 0
0028 0
0029 0
0030 0
0031 0
0032 0
0033 0
0034 0
0035 0
0036 0
0037 0
0038 0
0039 0
0040 0
0041 0
0042 0
0043 0
0044 0
0045 0
0046 0
0047 0
0048 0
0049 0
0050 0
0051 0
0052 0
0053 0
0054 0
0055 0
0056 0
0057 0

```
*****  
*  
* 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.  
*  
*****
```

* This file, EDT.REQ, contains definitions for EDT.

Edit history:

- 1-001 - Beginning of edit history.
- 1-002 - Add ASSERT macro, remove bugcheck codes. JBS 01-Jun-1981
- 1-003 - Offset the PDP-11 error codes, so they can be distinguished
from system-specific error codes. JBS 16-Jul-1981
- 1-004 - Remove the error messages, putting them in
ERRMSG.REQ. JBS 20-Jul-1981
- 1-005 - Add two fields to TBCB ; one points to the previous buffer,
the other marks the buffer as a macro. Delete
the creation of the MAC_BLOCK structure TMV 6-Aug-81
- 1-006 - Add the verb number for the new bell verb. STS 10-Aug-1981
- 1-007 - Add INP_JOURNAL and INP_COMMAND to replace INP_FILE. This
lets us journal the responses to SUBSTITUTE/QUERY in
the journal file. JBS 16-Aug-1981
- 1-008 - Add the verb number for the new day/time verb. STS 31-Aug-1981
- 1-009 - Update the routine and variable names. JBS & TMV 16-Aep-1981
- 1-010 - Add new verbs to set up default verb. STS 21-Sep-1981
- 1-011 - Add new verbs for delete select and toggle select. STS 23-Sep-1981
- 1-012 - Add new search and select verb. STS 24-Sep-1981
- 1-013 - Add literals for word and para types. STS 23-Oct-1981
- 1-014 - Add PREV_RANGE. JBS 02-Nov-1981
- 1-015 - Add definitions for file i/o codes and streams. STS 08-Dec-1981
- 1-016 - Change edt\$\$k to edt\$k for file i/o definitions. STS 09-Dec-1981
- 1-017 - Add macro to set up address and length in string desc. STS 11-Jan-1982
- 1-018 - Fix above macro to work with 11's. STS 13-Jan-1982
- 1-019 - Add literals for open output seq and open output noseq. STS 13-Jan-1982
- 1-020 - Chang string desc macro for bliss16. STS 15-Jan-1982
- 1-021 - Change 32-bit arithmetic to 48-bit arithmetic. SMB 15-Jan-1982
- 1-022 - Modify block allocation so that odd address traps don't occur on 11's. SMB 25-Jan-1982
- 1-023 - Remove original line numbers. SMB 29-Jan-1982

```
0058 0 1-024 - Make callable literals global. STS 08-Mar-1982
0059 0 1-025 - Remove callable literals. STS 08-Mar-1982
0060 0 1-026 - Add symbols for control C handling. JBS 24-May-1982
0061 0 1-027 - Change VMS multiply. SMB 25-May-1982
0062 0 1-028 - Add EDT$$K_FMT_BUFLen. JBS 05-Jul-1982
0063 0 1-029 - Add verb for xlate. STS 13-Aug-1982
0064 0 1-030 - Remove the keypad definitions to KEYPADDEF.REQ. JBS 13-Aug-1982
0065 0 1-031 - Add ASC_K_CSI, for 8-bit keyboards. JBS 17-Aug-1982
0066 0 1-032 - Add ASC_K_SS3, for 8-bit keyboards. JBS 20-Aug-1982
0067 0 1-033 - Add verb K_cls. STS 26-Aug-1982
0068 0 1-034 - Add K_RD^HED_LEN. JBS 31-Aug-1982
0069 0 1-035 - Add new screen data structures. SMB 11-Sep-1982
0070 0 1-036 - Put back a line that was deleted by mistake. SMB 15-Sep-1982
0071 0 1-037 - Revise the EDIT section of the new screen data structures. JBS 17-Sep-1982
0072 0 1-038 - Add CC_RDCNT. JBS 17-Sep-1982
0073 0 1-039 - Remove CC_RDCNT. STS 20-Sep-1982
0074 0 1-040 - Work on conditionalizing addline macro for speed. STS 30-Sep-1982
0075 0 1-041 - Add memory allocation maximum. SMB 18-Oct-1982
0076 0 1-042 - Add macros for comparing line numbers. STS 20-Oct-1982
0077 0 1-043 - Work on 11-version of compare macro. STS 21-Oct-1982
0078 0 1-044 - Bind high word of linenumbers in compare macro. STS 21-Oct-1982
0079 0 1-045 - Fix bug in compare. STS 22-Oct-1982
0080 0 1-046 - Work on 11 version of compare macro. STS 26-Oct-1982
0081 0 1-047 - Change 11 compare to call EDT$$CMP_LNO. STS 27-Oct-1982
0082 0 1-048 - Add SCR_EDIT_MINPOS, remove a bunch of unused and obsolete definitions. JBS 27-Oct-1982
0083 0 1-049 - Reduce the size of the screen edit area on the PDP-11. This saves
0084 0 space at the expense of time. JBS 15-Nov-1982
0085 0 1-050 - Remove the edit buffer entirely. JBS 27-Dec-1982
0086 0 1-051 - Reduce the amount of code generated by the ASSERT macro, to try
0087 0 to save space on the PDP-11. JBS 16-Jan-1983
0088 0 1-052 - Correct the definition of SS3. JBS 19-Jan-1983
0089 0 1-053 - Change the format buffer size for VMS. SMB 24-Feb-1983
0090 0 1-054 - Remove WC_K_NUM_BUKT. JBS 29-Mar-1983
0091 0
```

```
0092 0 | +
0093 0 | DEFINITION_DEFINITIONS
0094 0 |
0095 0 |
0096 0 | - The following definitions are used to facilitate further definitions.
0097 0 | -
0098 0 |
0099 0 | +
0100 0 | Field definition macros. This set of macros allows for definitions
0101 0 | of the fields of data structures, letting the compiler compute the
0102 0 | the offsets.
0103 0 | -
0104 0 |
0105 0 | COMPILETIME FIELD_OFFSET = 0;
0106 0 | COMPILETIME NUMBER_ONE = 1;
0107 0 |
M 0108 0 | MACRO START_FIELDS(FIELD_NAME) =
M 0109 0 |   FIELD FIELD_NAME =
M 0110 0 |   SET
0111 0 |   %ASSIGN(FIELD_OFFSET,0) %;
0112 0 |
M 0113 0 | MACRO A_FIELD(FIELD_NAME1,LENGTH) =
M 0114 0 |   FIELD_NAME1 = [FIELD_OFFSET/8,FIELD_OFFSET MOD 8,LENGTH,0]
0115 0 |   %ASSIGN(FIELD_OFFSET,FIELD_OFFSET+LENGTH) %;
0116 0 |
M 0117 0 | MACRO INC_FIELD (LENGTH) =
0118 0 |   %ASSIGN(FIELD_OFFSET,FIELD_OFFSET+LENGTH) %;
0119 0 |
0120 0 | MACRO END_FIELDS = TES;%;
0121 0 |
0122 0 | MACRO STRUC_SIZE(SIZE) = LITERAL SIZE = (FIELD_OFFSET+7)/8; %;
```

0123 0
0124 0
0125 0
0126 0
0127 0
0128 0
0129 0
0130 0
0131 0
0132 0
0133 0
0134 0
0135 0

+ IMPLEMENTATION PARAMETERS.

The following definitions are parameters used in the work-file system which may require re-definition for different implementations.

LITERAL
WF_BLN_LEN = 16; ! Bit length of a work-file block number.
LINE_NUM_LEN = 16; ! Bit length of a line number. (actually 3*16=48)

0136 0
0137 0
0138 0
0139 0
0140 0
0141 0
0142 0
0143 0
0144 0
0145 0
0146 0
0147 0
0148 0
0149 0
0150 0
0151 0
0152 0
0153 0
0154 0
0155 0
0156 0
0157 0
0158 0
0159 0
0160 0
0161 0
0162 0
0163 0
0164 0
0165 0
0166 0
0167 0
0168 0
0169 0
0170 0
0171 0
0172 0
0173 0
0174 0
0175 0
0176 0
0177 0
0178 0
0179 0
0180 0
0181 0
0182 0
0183 0
0184 0
0185 0
0186 0
0187 0
0188 0
0189 0
0190 0
0191 0
0192 0

↑
TBCB_DEFINITION

The EDT work file can contain multiple, independent data sets referred to as Text Buffers. A text buffer corresponds to the construct of the same name found in EDT user documentation, it is a sequential file of variable length records. The records are grouped together into blocks of 512 characters. The records in a block are sequentially ordered, though the blocks themselves are not. Each block contains a two-byte link to the previous and following blocks. In addition to the lines in the work-file, an input file may be associated with a text buffer. In this case the input file is logically placed at the end of the text buffer. The Text buffer is accessed via a control block called the Text Buffer Control Block, or TBCB.

START_FIELDS(TBCB_FIELDS)
A_FIELD(TBCB_LINE_ADDR,%BPADDR), ! Pointer to current line.
A_FIELD(TBCB_CUR_BUKT,WF_BLN_LEN), ! Current bucket number.
A_FIELD(TBCB_CUR_LIN,LINE_NUM_LEN), ! Current line number.
A_FIELD(TBCB_CUR_LINM,LINE_NUM_LEN),
A_FIELD(TBCB_CUR_LINH,LINE_NUM_LEN),
A_FIELD(TBCB_CHAR_POS,WF_BLN_LEN), ! The character position within the line
A_FIELD(TBCB_FIRST_BUKT,WF_BLN_LEN), ! First bucket number.
A_FIELD(TBCB_LAST_BUKT,WF_BLN_LEN), ! Last bucket number.
A_FIELD(TBCB_INPUT_LIN,LINE_NUM_LEN), ! Number of last input line.
A_FIELD(TBCB_INPUT_LINM,LINE_NUM_LEN),
A_FIELD(TBCB_INPUT_LINH,LINE_NUM_LEN),
A_FIELD(TBCB_LINE_COUNT,LINE_NUM_LEN), ! Count of lines in buffer.
A_FIELD(TBCB_LC_M,LINE_NUM_LEN),
A_FIELD(TBCB_LC_H,LINE_NUM_LEN),
A_FIELD(TBCB_CHAR_COUNT,%BPVAL), ! Count of chars in buffer.
A_FIELD(TBCB_PREV_BUF,%BPADDR), ! Pointer to previous text buffer.
A_FIELD(TBCB_NEXT_BUF,%BPADDR), ! Pointer to next text buffer.
A_FIELD(TBCB_INPUT_RAB,8), ! Pointer to input RAB.
A_FIELD(TBCB_IS_MAC,8), ! This buffer is a macro
A_FIELD(TBCB_NAME_LEN,8), ! Length of buffer name.
A_FIELD(TBCB_NAME,0) ! Name of buffer

END_FIELDS

STRUC_SIZE(TBCB_SIZE) ! Define size of TBCB.

MACRO TBCB_BLOCK = BLOCK[TBCB_SIZE,BYTE] FIELD(TBCB_FIELDS)% ;

↑
The pos block is the portion of the TBCB which contains information needed to locate the current line. This block must be identical to the first part of the TBCB or everything will fail.

START_FIELDS(POS_FIELDS)
A_FIELD(POS_LINE_ADDR,%BPADDR), ! Pointer to current line.
A_FIELD(POS_CUR_BUKT,WF_BLN_LEN), ! Current bucket number.
A_FIELD(POS_CUR_LIN,LINE_NUM_LEN), ! Current line number.

N 2
15-Sep-1984 23:00:56
15-Sep-1984 22:43:32

VAX-11 Bliss-32 V4.0-742
_S255\$DUA28:[EDT.SRC]EDT.REQ;1

Page (4) 6

```
0193 0      A_FIELD(POS_CUR_LINM,LINE_NUM_LEN),  
0194 0      A_FIELD(POS_CUR_LINH,LINE_NUM_LEN),  
0195 0      A_FIELD(POS_CHAR_POS,WF_BCN_LEN)  
0196 0      END_FIELDS  
0197 0  
0198 0      STRUC_SIZE(POS_SIZE)          ! Define size of position information  
0199 0  
0200 0      MACRO POS_BLOCK = BLOCK[POS_SIZE,BYTE] FIELD(POS_FIELDS)%;
```

```
0201 0
0202 0
0203 0 | +
0204 0 | TEXT LINE DEFINITIONS
0205 0 | A line number contains an integer part and a fractional part.
0206 0 | -
0207 0
0208 0 START_FIELDS(LIN_FIELDS)
0209 0 A_FIELD(LIN_LENGTH,8) ! Length of line
0210 0 A_FIELD(LIN_NUM,LINE_NUM_LEN) ! The line number
0211 0 A_FIELD(LIN_NUMM,LINE_NUM_LEN),
0212 0 A_FIELD(LIN_NUMH,LINE_NUM_LEN),
0213 0 A_FIELD(LIN_TEXT,0) ! The actual text
0214 0 END_FIELDS
0215 0
0216 0 STRUC_SIZE(LIN_FIXED_SIZE)
0217 0
0218 0 MACRO LIN_BLOCK = BLOCK[LIN_FIXED_SIZE, BYTE] FIELD(LIN_FIELDS)%;
```

```
0219 0
0220 0
0221 0
0222 0
0223 0
0224 0
0225 0
0226 0
0227 0
0228 0
0229 0
0230 0
0231 0
0232 0
0233 0
0234 0
0235 0
0236 0
0237 0

+
WORK-FILE_BUCKET_DEFINITIONS
-
The work file is organized into blocks of WF_BLOCK_SIZE characters.
Each Text Buffer in the work file consists of a linked list of blocks.

LITERAL WF_BUKT_SIZE = 512;           ! Size of a work-file block

START_FIELDS(WFB_FIELDS)
  A_FIELD(WFB_PREV_BUKT,WF_BLN_LEN),  ! Number of previous bucket
  A_FIELD(WFB_NEXT_BUKT,WF_BLN_LEN),  ! Number of next bucket
  A_FIELD(WFB_END,%BPVAL),            ! Offset to last record in block
  A_FIELD(WFB_RECORDS,0)              ! Address of first record in block
END_FIELDS

STRUC_SIZE(WFB_FIXED_SIZE)
```

```
0238 0  
0239 0  
0240 0  
0241 0  
0242 0  
0243 0  
0244 0  
0245 0  
0246 0  
0247 0  
0248 0  
0249 0  
0250 0  
0251 0  
0252 0  
0253 0  
0254 0  
0255 0  
0256 0  
0257 0  
0258 0  
+  
LINE NUMBER BLOCK DEFINITIONS  
-  
The line number is defined as a block, so it can be handled as  
three 16-bit words.  
FIELD LN_FIELDS =  
  SET  
  LN_LO = [0,0,16,0],  
  LN_MD = [2,0,16,0],  
  LN_HI = [4,0,16,0]  
  TES;  
MACRO LN_BLOCK = BLOCK[6,BYTE] FIELD(LN_FIELDS) %;  
LITERAL LN_SIZE = 6;  
STRUCTURE  
  LNOVECTOR[I;N] = [N*LN_SIZE] (LNOVECTOR+I*LN_SIZE);
```

0259 0
 0260 0
 0261 0
 0262 0
 0263 0
 0264 0
 0265 0
 0266 0
 0267 0
 0268 0
 0269 0
 0270 0
 0271 0
 0272 0
 0273 0
 0274 0
 0275 0
 0276 0
 0277 0
 0278 0
 0279 0
 0280 0
 0281 0
 0282 0
 0283 0
 0284 0
 0285 0
 0286 0
 0287 0
 0288 0
 0289 0
 0290 0
 0291 0
 0292 0
 0293 0
 0294 0
 0295 0
 0296 0
 0297 0
 0298 0
 0299 0
 0300 0
 0301 0
 0302 0
 0303 0
 0304 0
 0305 0
 0306 0
 0307 0
 0308 0
 0309 0
 0310 0
 0311 0
 0312 0
 0313 0
 0314 0
 0315 0

+ Semantic node definitions.
 The following defines the structures created by the EDT
 command parser semantic routines. These structures form
 a tree-like representation of the command.
 The fields which are grouped together are re-definitions of the
 same slot for use in different types of nodes.

```
FIELD NODE_FIELDS =
  SET
  NODE_TYPE      = [0,0,8,0],           ! Identifies the type of node
  COM_NUM        = [1,0,8,0],           ! Identifies the command
  RAN_TYPE       = [1,0,8,0],           ! Identifier type of range
  OP_TYPE        = [1,0,8,0],           ! Identifies type of operand
  SEQ_VAL        = [1,0,8,0],           ! Did the seq switch have value.
  RANGE1         = [%UPVAL,0,%BPVAL,0], ! First range specifier
  RAN_VAL        = [%UPVAL,0,%BPVAL,0], ! Value for range specifier
  SW_BITS        = [%UPVAL,0,%BPVAL,0], ! Bits for each possible switch
  SRCHADDR       = [%UPVAL,0,%BPVAL,0], ! Address of search string
  SET_TYPE       = [%UPVAL,0,%BPVAL,0], ! Which type of set command
  LEFT_OP        = [%UPVAL,0,%BPVAL,0], ! Left operand for binary ops
  OP_LEN         = [%UPVAL,0,%BPVAL,0], ! operand length for op nodes.
  OP_VAL         = [%UPVAL,0,%BPVAL,0], ! Operand value for numerics.
  COM_EXPR       = [%UPVAL,0,%BPVAL,0], ! Expression pointer for LET
  OP_LEFTOP      = [%UPVAL,0,%BPVAL,0], ! Left operand for operators.
  SUB_BASE       = [%UPVAL,0,%BPVAL,0], ! Substring base string.
  RANGE2         = [%UPVAL*2,0,%BPVAL,0], ! Second range specifier
  SUB_RANGE      = [%UPVAL*2,0,%BPVAL,0], ! Pointer to range for ranges
  STR_PNT        = [%UPVAL*2,0,%BPVAL,0], ! Pointer to a search string
  SRCHLEN        = [%UPVAL*2,0,%BPVAL,0], ! Search string length
  FILSPEC        = [%UPVAL*2,0,%BPVAL,0], ! File specification address
  SW_VAL1        = [%UPVAL*2,0,%BPVAL,0], ! First value for switches
  AS_STR         = [%UPVAL*2,0,%BPVAL,0], ! Addr of string for AS
  RIGHT_OP       = [%UPVAL*2,0,%BPVAL,0], ! Right operand for binary ops.
  BUF_NAME       = [%UPVAL*2,0,%BPVAL,0], ! Address of buffer name
  OP_ADDR        = [%UPVAL*2,0,%BPVAL,0], ! Operand address for op nodes.
  COM_VARBL      = [%UPVAL*2,0,%BPVAL,0], ! Variable pointer for LET
  OP_RIGHTOP     = [%UPVAL*2,0,%BPVAL,0], ! Right operand for operators.
  SUB_START      = [%UPVAL*2,0,%BPVAL,0], ! Substring start pos.
  TAB_COUNT      = [%UPVAL*2,0,%BPVAL,0], ! Count for tabs adjust.
  SET_VAL1       = [%UPVAL*3,0,%BPVAL,0], ! Value for set command
  REPADDR        = [%UPVAL*3,0,%BPVAL,0], ! Replace string address
  FSPCLN         = [%UPVAL*3,0,%BPVAL,0], ! File spec length
  AS_LEN         = [%UPVAL*3,0,%BPVAL,0], ! Length of string for AS
  BUF_LEN        = [%UPVAL*3,0,%BPVAL,0], ! length of buffer name
  SUB_LENGTH     = [%UPVAL*3,0,%BPVAL,0], ! Substring length.
  NEXT_COM       = [%UPVAL*4,0,%BPVAL,0], ! Pointer to next command
  NEXT_RANGE     = [%UPVAL*4,0,%BPVAL,0], ! Pointer to next range
```

```
0316 0 REPLEN = [%UPVAL*4,0,%BPVAL,0], ! Replace string length
0317 0 SET_VAL = [%UPVAL*4,0,%BPVAL,0],
0318 0 KEY_VAL = [%UPVAL*4,0,%BPVAL,0], ! Number of key for def key
0319 0
0320 0 PREV_RANGE = [%UPVAL*5,0,%BPVAL,0], ! Reverse of NEXT_RANGE
0321 0 SWITS = [%UPVAL*5,0,%BPVAL,0], ! Switch block pointer
0322 0 SW_VAL2 = [%UPVAL*5,0,%BPVAL,0], ! Second option switch value
0323 0
0324 0 SW_OVR1 = [%UPVAL*6,0,%BPVAL,0], ! Part of second option switch
0325 0
0326 0 SW_OVR2 = [%UPVAL*7,0,%BPVAL,0] ! Part of second option switch
0327 0 TES;
0328 0
0329 0 LITERAL
0330 0 NUM_NODES = 20, ! Number of semantic nodes
0331 0 NODE_SIZE = 8*%UPVAL; ! Size of semantic node
0332 0
0333 0 LITERAL ! Node type equates
0334 0
0335 0 COM_NODE = 1, ! Command node
0336 0 RANGE_NODE = 2, ! Range node
0337 0 STR_NODE = 3, ! SUBSTITUTE strings
0338 0 SW_NODE = 4, ! Option switch value
0339 0 OP_NODE = 5; ! Expression operand
0340 0
0341 0 MACRO NODE_BLOCK = BLOCK[NODE_SIZE, BYTE] FIELD(NODE_FIELDS) %;
```

0342 0
0343 0
0344 0
0345 0
0346 0
0347 0
0348 0
0349 0
0350 0
0351 0
0352 0
0353 0
0354 0
0355 0
0356 0
0357 0
0358 0
0359 0
0360 0
0361 0
0362 0
0363 0

!+ ASCII CHARACTER DEFINITIONS
!- Commonly used non-printing ASCII characters.

LITERAL
ASC_K_BS = %0'10',
ASC_K_TAB = %0'11',
ASC_K_LF = %0'12',
ASC_K_CTRL_K = %0'13',
ASC_K_FF = %0'14',
ASC_K_CR = %0'15',
ASC_K_SO = %0'16',
ASC_K_SI = %0'17',
ASC_K_CTRL_U = %0'25',
ASC_K_CTRL_Z = %0'32',
ASC_K_ESC = %0'33',
ASC_K_SP = %0'40',
ASC_K_DEL = %0'177',
ASC_K_CSI = ASC_K_ESC + %X'80',
ASC_K_SS3 = ASC_K_SI + %X'80';

0364 0
0365 0
0366 0
0367 0
0368 0
0369 0
0370 0
0371 0
0372 0
0373 0
0374 0
0375 0
0376 0
0377 0
0378 0
0379 0
0380 0
0381 0
0382 0
0383 0
0384 0
0385 0
0386 0
0387 0
0388 0
0389 0
0390 0
0391 0
0392 0
0393 0
0394 0
0395 0
0396 0
0397 0
0398 0
0399 0

!+
COMMAND NUMBER DEFINITIONS
The following values are used in a command type node to specify which
command it is.
-

LITERAL
COM_NULL = 0.
COM_CHANGE = 1.
COM_COPY = 2.
COM_DEFINE = 3.
COM_DELETE = 4.
COM_EXIT = 5.
COM_FIND = 6.
COM_INCLUDE = 7.
COM_INSERT = 8.
COM_MOVE = 9.
COM_PRINT = 10.
COM_QUIT = 11.
COM_REPLACE = 12.
COM_RESEQ = 13.
COM_SET = 14.
COM_SHOW = 15.
COM_SUBS = 16.
COM_TYPE = 17.
COM_WRITE = 18.
COM_SUBS_NEXT = 19.
COM_HELP = 20.
COM_CLEAR = 21.
COM_TADJ = 22.
COM_FILL = 23.
COM_DEF_MAC = 24.
COM_MAC_CALL = 25.
! COM_VERIFY = ?.
LAST_COM = 25;

0400 0
0401 0
0402 0
0403 0
0404 0
0405 0
0406 0
0407 0
0408 0
0409 0
0410 0
0411 0
0412 0
0413 0
0414 0
0415 0
0416 0
0417 0
0418 0
0419 0
0420 0
0421 0
0422 0
0423 0
0424 0
0425 0
0426 0
0427 0
0428 0
0429 0
0430 0
0431 0
0432 0
0433 0
0434 0
0435 0
0436 0
0437 0
0438 0
0439 0
0440 0
0441 0
0442 0
0443 0
0444 0
0445 0
0446 0
0447 0
0448 0
0449 0
0450 0
0451 0
0452 0
0453 0
0454 0
0455 0
0456 0

↑
RANGE TYPE DEFINITIONS

The following constants are used in range nodes to specify the type of range.

LITERAL

RAN_NULL = 0,
RAN_NUMBER = 1,
RAN_DOT = 2,
RAN_STR = 3,
RAN_BEGIN = 4,
RAN_END = 5,
RAN_ORIG = 6,
RAN_PATTERN = 7,
RAN_LAST = 8,
RAN_BEFORE = 9,
RAN_REST = 10,
RAN_WHOLE = 11,
RAN_SELECT = 12,
RAN_BUFFER = 13,
RAN_PLUS = 14,
RAN_MINUS = 15,
RAN_FOR = 16,
RAN_THRU = 17,
RAN_MINSTR = 18,
RAN_ALL = 19,
RAN_AND = 20,
NUM_RAN = 20, ! Total number of ranges
NUM_SLR = 7; ! number of single line ranges

Operand types for operand nodes.

LITERAL

OP_STRING = 0, ! Operand is a quoted string
OP_NUM = 1, ! Operand is a number
OP_VAR = 2, ! Operand is a variable
OP_DOT = 3, ! Operand is the dot pseudo variable
OP_ADD = 4, ! Operand is an addition operator
OP_SUB = 5, ! Operand is a subtractions operator
OP_MULT = 6, ! Operand is a multiplication operator
OP_DIV = 7, ! Operand is a division operator
OP_AND = 8, ! logical and
OP_OR = 9, ! logical or
OP_LSS = 10, ! compare for less
OP_LEQ = 11, ! compare for less or equal
OP_EQL = 12, ! Compare for equality
OP_GEQ = 13, ! compare for greater or equal
OP_GTR = 14, ! compare for greater
OP_NEQ = 15, ! compare for not equal
OP_AMP = 16, ! concatenation
OP_SUBSTR = 17, ! substringing
OP_NEG = 18, ! negation
OP_NOT = 19, ! logical not

J 3
15-Sep-1984 23:00:56
15-Sep-1984 22:43:32

VAX-11 Bliss-32 V4.0-742
_S255\$DUA28:[EDT.SRC]EDT.REQ;1

Page 15
(11)

```
: 0457 0      OP_LENGTH = 20,  ! length of
: 0458 0      OP_COL   = 21,  ! current column
: 0459 0      OP_FIND  = 22,
: 0460 0      OP_POS   = 23,  ! current position
: 0461 0      OP_LAST_OP = 23; ! last operand type
```

```
0462 0  | +
0463 0  | LINE NUMBER HANDLING MACROS
0464 0  |
0465 0  | These macros are used for arithmetic involving line numbers, so it can
0466 0  | be transportable across systems with various word lengths. At least 48
0467 0  | bits of precision are required for line numbers. Line numbers are stored
0468 0  | as an integer with a scale of -5, i.e. the true value * 10**5, so we can
0469 0  | have 5 decimal positions and 10 integer positions in the line number.
0470 0  | -
0471 0  |
0472 0  | %IF %BLISS(BLISS32) %THEN
0473 0  | MACRO
M 0474 0  |     ADDLINE(S1,S2,DEST,MAX) =
M 0475 0  | | +
M 0476 0  | | Add 2 48-bit numbers using 2 longwords (so we can
M 0477 0  | | use the BLISS-32 Built-in macros).
M 0478 0  | | -
M 0479 0  | | BEGIN
M 0480 0  | | %IF %CTCE(S1) %THEN
M 0481 0  | |     %IF %LENGTH EQL 2 %THEN
M 0482 0  | | | +
M 0483 0  | | | add a compile time expression to s2 and store it in s2
M 0484 0  | | | -
M 0485 0  | | | BEGIN
M 0486 0  | | | BIND
M 0487 0  | | |     FIRST_LWORD = S2 :LONG,
M 0488 0  | | |     NEXT_WORD = (S2+4) : WORD;
M 0489 0  | | |     FIRST_LWORD = .FIRST_LWORD + S1;
M 0490 0  | | |     IF .FIRST_LWORD LSSU S1
M 0491 0  | | |     THEN
M 0492 0  | | |     NEXT_WORD = .NEXT_WORD + 1;
M 0493 0  | | | END
M 0494 0  | | | %ELSE
M 0495 0  | | | +
M 0496 0  | | | add a compile time expression to s2 and store it in dest
M 0497 0  | | | -
M 0498 0  | | | BEGIN
M 0499 0  | | | BIND FIRST_WORD = (DEST) : LONG,
M 0500 0  | | |     NEXT_WORD = (DEST+4) : WORD,
M 0501 0  | | |     SOURCE_2LO = (S2) : LONG,
M 0502 0  | | |     SOURCE_2HI = (S2+4) : WORD;
M 0503 0  | | |
M 0504 0  | | |     FIRST_WORD = .SOURCE_2LO + S1;
M 0505 0  | | |     IF (.FIRST_WORD LSSU S1)
M 0506 0  | | |     THEN
M 0507 0  | | |     NEXT_WORD = .SOURCE_2HI + 1
M 0508 0  | | |     ELSE
M 0509 0  | | |     NEXT_WORD = .SOURCE_2HI;
M 0510 0  | | | END
M 0511 0  | | | %FI
M 0512 0  | | | %ELSE
M 0513 0  | | | +
M 0514 0  | | | we don't have a compile time expression, but we are adding two 48-bit numbers
M 0515 0  | | | -
M 0516 0  | | |
M 0517 0  | | |     %IF %LENGTH EQL 2 %THEN
M 0518 0  | | |     BEGIN
M 0518 0  | | |         ! store the result in S2
M 0518 0  | | |     END
```

```
.. M 0519 0 LOCAL SAVE: WORD;
M 0520 0 BUILTIN ADDM;
M 0521 0 BIND UPPER_WORD = (S2+6) : WORD;
M 0522 0 SAVE = .UPPER_WORD;
M 0523 0 ADDM(2,S1,S2,S2);
M 0524 0 UPPER_WORD = .SAVE;
M 0525 0 END
M 0526 0 %ELSE
M 0527 0 %IF %LENGTH EQL 3 %THEN ! store the result in DEST
M 0528 0 BEGIN
M 0529 0 LOCAL
M 0530 0 SAVE : WORD;
M 0531 0 BUILTIN ADDM;
M 0532 0 BIND UPPER_WORD = (DEST+6) : WORD;
M 0533 0
M 0534 0 SAVE = .UPPER_WORD;
M 0535 0 ADDM(2,S1,S2,DEST);
M 0536 0 UPPER_WORD = .SAVE;
M 0537 0 END
M 0538 0 %ELSE ! store the result in DEST and return
M 0539 0 BEGIN ! any overflow in MAX
M 0540 0 LOCAL
M 0541 0 SAVES2 : WORD,
M 0542 0 SAVED : WORD;
M 0543 0 BIND
M 0544 0 S1_UP = (S1+6) : WORD,
M 0545 0 S2_UP = (S2+6) : WORD,
M 0546 0 DEST_UP = (DEST+6) : WORD;
M 0547 0
M 0548 0 BUILTIN ADDM;
M 0549 0 SAVES2 = .S2_UP + .S1_UP;
M 0550 0 SAVED = .DEST_UP;
M 0551 0 ADDM(2,S1,S2,DEST);
M 0552 0
M 0553 0 !+
M 0554 0 !- Get the overflow bit
M 0555 0 !-
M 0556 0 IF .DEST_UP EQL .SAVES2
M 0557 0 THEN
M 0558 0 MAX = 0
M 0559 0 ELSE
M 0560 0 MAX = 1;
M 0561 0 DEST_UP = .SAVED;
M 0562 0 END
M 0563 0 %FI
M 0564 0 %FI
M 0565 0 %FI
M 0566 0 END%,
M 0567 0
M 0568 0
M 0569 0 SUBLINE(S1,S2,DEST) =
M 0570 0 !+
M 0571 0 !- Subtract 2 48-bit numbers using 2 longwords
M 0572 0 !-
M 0573 0 BEGIN
M 0574 0 %IF %CTCE(S1) %THEN
M 0575 0 %IF %LENGTH EQL 2 %THEN
```

```

M 0576 0
M 0577 0
M 0578 0
M 0579 0
M 0580 0
M 0581 0
M 0582 0
M 0583 0
M 0584 0
M 0585 0
M 0586 0
M 0587 0
M 0588 0
M 0589 0
M 0590 0
M 0591 0
M 0592 0
M 0593 0
M 0594 0
M 0595 0
M 0596 0
M 0597 0
M 0598 0
M 0599 0
M 0600 0
M 0601 0
M 0602 0
M 0603 0
M 0604 0
M 0605 0
M 0606 0
M 0607 0
M 0608 0
M 0609 0
M 0610 0
M 0611 0
M 0612 0
M 0613 0
M 0614 0
M 0615 0
M 0616 0
M 0617 0
M 0618 0
M 0619 0
M 0620 0
M 0621 0
M 0622 0
M 0623 0
M 0624 0
M 0625 0
M 0626 0
M 0627 0
M 0628 0
M 0629 0
M 0630 0
M 0631 0
M 0632 0

+
- we have a compile time expression to add to S2 and store in S2
  BEGIN
  LOCAL SAVE : LONG;
  BIND
    FIRST_WORD = S2 : LONG,
    NEXT_WORD = (S2+4) : WORD;
  SAVE = .FIRST_WORD;
  FIRST_WORD = .FIRST_WORD - S1;
  IF .FIRST_WORD GTRU .SAVE
  THEN
    NEXT_WORD = .NEXT_WORD - 1;
  END
  %ELSE
+
- add the compile time expression to S2 and store it in DEST
  BEGIN
  BIND FIRST_WORD = (DEST) : LONG,
        NEXT_WORD = (DEST+4) : WORD,
        SOURCE_2LO = (S2) : LONG,
        SOURCE_2HI = (S2+4) : WORD;

  FIRST_WORD = .SOURCE_2LO - S1;
  IF .FIRST_WORD GTRU .SOURCE_2LO
  THEN
    NEXT_WORD = .SOURCE_2HI - 1
  ELSE
    NEXT_WORD = .SOURCE_2HI;
  END
  %FI
  %ELSE
  %IF %LENGTH EQL 2 %THEN
+
- add two 48 bit numbers and store result in S2
  BEGIN
  LOCAL SAVE : WORD;
  BUILTIN SUBM;
  BIND UPPER_WORD = (S2+6) : WORD;
  SAVE = .UPPER_WORD;
  SUBM(2,S1,S2,S2);
  UPPER_WORD = .SAVE;
  END
  %ELSE
+
- add two 48 bit numbers and store result in DEST
  BEGIN
  LOCAL
    SAVE : WORD;
  BUILTIN SUBM;
  BIND UPPER_WORD = (DEST+6) : WORD;

  SAVE = .UPPER_WORD;
  SUBM(2,S1,S2,-DEST);
```

0
M 0633 0
M 0634 0
M 0635 0
M 0636 0
0637 0
0638 0
0639 0
M 0640 0
M 0641 0
M 0642 0
M 0643 0
M 0644 0
M 0645 0
M 0646 0
M 0647 0
M 0648 0
M 0649 0
M 0650 0
M 0651 0
M 0652 0
M 0653 0
M 0654 0
M 0655 0
M 0656 0
M 0657 0
M 0658 0
M 0659 0
M 0660 0
M 0661 0
M 0662 0
M 0663 0
M 0664 0
M 0665 0
M 0666 0
M 0667 0
0668 0
0669 0
0670 0
0671 0
0672 0
M 0673 0
M 0674 0
M 0675 0
M 0676 0
M 0677 0
M 0678 0
M 0679 0
M 0680 0
M 0681 0
M 0682 0
M 0683 0
M 0684 0
M 0685 0
M 0686 0
M 0687 0
0688 0
0689 0

```
UPPER_WORD = .SAVE;  
END  
%FI  
%FI  
END%,  
  
MULTLINE(S1,S2,DEST) =  
+ Multiply 2 48-bit numbers, but S1 MUST be <= 100,000  
-  
  BEGIN  
  BIND  
    M1 = S1 : BITVECTOR [32];  
  
    LOCAL M2 : VECTOR[2],  
          P : VECTOR[2];  
    BUILTIN ADDM, ASHQ;  
  
+ Set up the multiplicand and result in 64 bits, zeroeing  
- out the upper 16-bits.  
  M2[0] = .(S2)<0,32>; M2[1] = .(S2+4)<0,16>;  
  P[0] = 0; P[1] = 0;  
  
+ Since 65535 < multiplier <+ 100,000... we only need to  
- examine the low order 17-bits.  
  DECR I FROM 16 TO 0  
  DO  
    BEGIN  
      ASHQ(%REF(1), P, P);           ! Shift result left by 1 (multiply by 2)  
      IF (.M1[I]) THEN ADDM(2, P, M2, P); ! Add multiplicand to result  
      END;                          ! if multiplier bit set  
      (DEST)<0,32> = .P[0]; (DEST+4)<0,16> = .P[1];  
    END%,  
  
+ compare two 48 bit line numbers to see if they are equal  
-  
  LINNOEQL(LIN1,LIN2) =  
  BEGIN  
  BIND  
    NO_1 = LIN1 : VECTOR[3,WORD],  
    NO_2 = LIN2 : VECTOR[3,WORD],  
    LOW_1 = NO_1[0] : LONG,  
    LOW_2 = NO_2[0] : LONG,  
    HIGH_1 = NO_1[2] : WORD,  
    HIGH_2 = NO_2[2] : WORD;  
  
    IF ((.LOW_1 EQL .LOW_2) AND (.HIGH_1 EQL .HIGH_2))  
    THEN  
      (1)  
    ELSE  
      (0)  
    END%,
```

```
.. M 0690 0      CMLPNO(LIN1,LIN2) =  
.. M 0691 0      BEGIN  
.. M 0692 0      BIND  
.. M 0693 0      NO_1 = LIN1 : VECTOR[3,WORD],  
.. M 0694 0      NO_2 = LIN2 : VECTOR[3,WORD],  
.. M 0695 0      LOW_1 = NO_1[0] : LONG,  
.. M 0696 0      LOW_2 = NO_2[0] : LONG,  
.. M 0697 0      HIGH_1 = NO_1[2] : WORD,  
.. M 0698 0      HIGH_2 = NO_2[2] : WORD;  
.. M 0699 0  
.. M 0700 0      IF (.HIGH_1 LSSU .HIGH_2)  
.. M 0701 0      THEN  
.. M 0702 0      (-1)  
.. M 0703 0      ELSE  
.. M 0704 0      BEGIN  
.. M 0705 0      IF (.HIGH_1 EQL .HIGH_2)  
.. M 0706 0      THEN  
.. M 0707 0      IF (.LOW_1 LSSU .LOW_2)  
.. M 0708 0      THEN  
.. M 0709 0      (-1)  
.. M 0710 0      ELSE  
.. M 0711 0      IF (.LOW_1 EQL .LOW_2) THEN (0) ELSE (1)  
.. M 0712 0      ELSE  
.. M 0713 0      (1)  
.. M 0714 0      END  
.. M 0715 0      END%,  
.. M 0716 0  
.. M 0717 0      MOVELINE(S,D) = (CH$MOVE(6,S,D))%,           ! Move 6 bytes of storage  
.. M 0718 0  
.. M 0719 0      BUILDLINE(S,D) = (D = S; (D+4) = 0)%;       ! Build a number  
.. M 0720 0  
.. M 0721 0      !  
.. U 0722 0      %ELSE %IF %BLISS(BLISS16) %THEN  
.. U 0723 0  
.. U 0724 0      MACRO  
.. U 0725 0      ADDLINE(S1,S2,DEST,MAX) =  
.. U 0726 0      BEGIN  
.. U 0727 0      %IF %CTCE(S1) %THEN  
.. U 0728 0      %IF %LENGTH EQL 2 %THEN  
.. U 0729 0      !+  
.. U 0730 0      ! we are adding a constant to source_2 and storing in source_2  
.. U 0731 0      !-  
.. U 0732 0      BEGIN  
.. U 0733 0      BIND  
.. U 0734 0      FIRST_WORD = S2:WORD,  
.. U 0735 0      NEXT_WORD = (S2+2) : WORD,  
.. U 0736 0      HIGH_WORD = (S2+4) : WORD;  
.. U 0737 0      FIRST_WORD = .FIRST_WORD + S1;  
.. U 0738 0      IF .FIRST_WORD EQL 0  
.. U 0739 0      THEN  
.. U 0740 0      BEGIN  
.. U 0741 0      NEXT_WORD = .NEXT_WORD + 1;  
.. U 0742 0      IF .NEXT_WORD EQL 0 THEN HIGH_WORD = .HIGH_WORD + 1;  
.. U 0743 0      END;  
.. U 0744 0      END  
.. U 0745 0      %ELSE  
.. U 0746 0      !+
```

```
U 0747 0 ! destination is DEST and we have a compile time constant
U 0748 0 !-
U 0749 0 BEGIN
U 0750 0 BIND
U 0751 0 SOURCE_1 = S2 : WORD,
U 0752 0 SOURCE_2 = (S2+2) : WORD,
U 0753 0 SOURCE_3 = (S2+4) : WORD,
U 0754 0 FIRST_WORD = DEST : WORD,
U 0755 0 NEXT_WORD = (DEST+2) : WORD,
U 0756 0 HIGH_WORD = (DEST+4) : WORD;
U 0757 0 FIRST_WORD = .SOURCE_1 + S1;
U 0758 0 NEXT_WORD = .SOURCE_2;
U 0759 0 HIGH_WORD = .SOURCE_3;
U 0760 0 IF .FIRST_WORD EQL 0
U 0761 0 THEN
U 0762 0 BEGIN
U 0763 0 NEXT_WORD = .NEXT_WORD + 1;
U 0764 0 IF .NEXT_WORD EQL 0
U 0765 0 THEN
U 0766 0 HIGH_WORD = .HIGH_WORD + 1 ;
U 0767 0 END;
U 0768 0 END
U 0769 0 %FI
U 0770 0 !+
U 0771 0 ! we don't have a constant
U 0772 0 !-
U 0773 0
U 0774 0 %ELSE
U 0775 0 %IF %LENGTH EQL 2 %THEN
U 0776 0 BEGIN EXTERNAL ROUTINE A48_ADD; A48_ADD(S1,S2,S2) END
U 0777 0 %ELSE
U 0778 0 %IF %LENGTH EQL 3 %THEN
U 0779 0 BEGIN EXTERNAL ROUTINE A48_ADD; A48_ADD(S1,S2,DEST) END
U 0780 0 %ELSE
U 0781 0 BEGIN EXTERNAL ROUTINE A48_ADD; MAX = A48_ADD(S1,S2,DEST) END
U 0782 0 %FI
U 0783 0 %FI
U 0784 0 %FI
U 0785 0 END%,
U 0786 0 SUBLINE(S1,S2,DEST) =
U 0787 0 BEGIN
U 0788 0 %IF %CTCE(S1) %THEN
U 0789 0 BEGIN
U 0790 0 %IF %LENGTH EQL 2 %THEN
U 0791 0 BEGIN
U 0792 0 LOCAL SAVE : WORD;
U 0793 0 BIND
U 0794 0 FIRST_WORD = S2 : WORD,
U 0795 0 NEXT_WORD = (S2+2) : WORD,
U 0796 0 HIGH_WORD = (S2+4) : WORD;
U 0797 0 SAVE = .FIRST_WORD;
U 0798 0 FIRST_WORD = .FIRST_WORD - S1;
U 0799 0 IF .FIRST_WORD GTRU .SAVE
U 0800 0 THEN
U 0801 0 BEGIN
U 0802 0 NEXT_WORD = .NEXT_WORD - 1;
U 0803 0 IF .NEXT_WORD EQL -1 THEN HIGH_WORD = .HIGH_WORD - 1;
```

U 0804 0
U 0805 0
U 0806 0
U 0807 0
U 0808 0
U 0809 0
U 0810 0
U 0811 0
U 0812 0
U 0813 0
U 0814 0
U 0815 0
U 0816 0
U 0817 0
U 0818 0
U 0819 0
U 0820 0
U 0821 0
U 0822 0
U 0823 0
U 0824 0
U 0825 0
U 0826 0
U 0827 0
U 0828 0
U 0829 0
U 0830 0
U 0831 0
U 0832 0
U 0833 0
U 0834 0
U 0835 0
U 0836 0
U 0837 0
U 0838 0
U 0839 0
U 0840 0
U 0841 0
U 0842 0
U 0843 0
U 0844 0
U 0845 0
U 0846 0
U 0847 0
U 0848 0
U 0849 0
U 0850 0
U 0851 0
U 0852 0
U 0853 0
U 0854 0
U 0855 0

```
END;  
%ELSE  
+ subtract a compile time constant to S2 and put result in DEST  
-  
  BEGIN  
  BIND  
  FIRST_WORD = DEST : WORD,  
  NEXT_WORD = (DEST+2) : WORD,  
  HIGH_WORD = (DEST+4) : WORD,  
  S2_LO = S2 : WORD,  
  S2_M = (S2+2) : WORD,  
  S2_HI = (S2+4) : WORD;  
  
  FIRST_WORD = .S2_LO - S1;  
  NEXT_WORD = .S2_M;  
  HIGH_WORD = .S2_HI;  
  IF .FIRST_WORD GTRU .S2_LO  
  THEN  
  BEGIN  
  NEXT_WORD = .NEXT_WORD - 1;  
  IF .NEXT_WORD EQL -1  
  THEN  
  HIGH_WORD = .HIGH_WORD - 1;  
  END;  
  END  
%FI  
END  
%ELSE  
+ We don't have a compile time expression  
-  
  %IF %LENGTH EQL 2 %THEN  
  BEGIN EXTERNAL ROUTINE A48_SUB; A48_SUB(S1,S2,S2) END  
  %ELSE  
  BEGIN EXTERNAL ROUTINE A48_SUB; A48_SUB(S1,S2,DEST) END  
  %FI  
%FI  
END%,  
MULTLINE(S5,S6,D3) =  
  BEGIN EXTERNAL ROUTINE A48_MUL; A48_MUL(S5,S6,D3) END %,  
LINNOEQL (LIN1,LIN2) = (CH$EQL(6,LIN1,6,LIN2))%,  
CMLNO (LIN1,LIN2) =  
  BEGIN EXTERNAL ROUTINE EDT$$CMP_LNO; EDT$$CMP_LNO(LIN1,LIN2) END %,  
MOVELINE(S11,D6) = (CH$MOVE(6,S11,D6))%,  
BUILDLINE(S12,D7) = (D7 = S12; (D7+2) = 0; (D7+4) = 0)%;  
%FI %FI
```

0856 0
0857 0
0858 0
0859 0
0860 0
0861 0
0862 0
0863 0
0864 0
0865 0
0866 0
0867 0
0868 0
0869 0
0870 0
0871 0
0872 0
0873 0
0874 0
0875 0
0876 0

!+ OPTION SWITCH BIT DEFINITIONS
!-

LITERAL
OPT_QUERY = 2.
OPT_BRIEF = 4.
OPT_NOTYP = 8.
OPT_SEQ = 16.
OPT_DUPL = 32.
OPT_SAVE = 64.
OPT_STAY = 128.

MACRO
OPB_QUERY = 1,1 %.
OPB_BRIEF = 2,1 %.
OPB_NOTYP = 3,1 %.
OPB_SEQ = 4,1 %.
OPB_DUPL = 5,1 %.
OPB_SAVE = 6,1 %.
OPB_STAY = 7,1 %;

```
0877 0 | Input source definitions.
0878 0 |
0879 0 | These constants define the source command line input.
0880 0 |
0881 0 LITERAL
0882 0     INP_TERM      = 0,           | Terminal
0883 0     INP_MACRO    = 1,           | A macro
0884 0     INP_COMMAND  = 2,           | The startup file
0885 0     INP_JOURNAL  = 3;          | The journal file (only during /RECOVER)
0886 0 |
0887 0 | +
0888 0 | Terminal type definitions.
0889 0 |
0890 0 | These literals define the type of terminal we are running on.
0891 0 |
0892 0 LITERAL
0893 0     TERM_UNKNOWN= 0,
0894 0     TERM_VT52   = 1,
0895 0     TERM_VT100  = 2,
0896 0     TERM_HCPY   = 3;
0897 0 | +
0898 0 | Length of the type-ahead buffer
0899 0 |
0900 0 LITERAL
0901 0     K_RDAHED_LEN = 32;
0902 0 |
0903 0 |
0904 0 | Editor mode definitions.
0905 0 |
0906 0 LITERAL
0907 0     CHANGE_MODE  = 0,
0908 0     LINE_MODE    = 1;
0909 0 |
0910 0 |
0911 0 | definitions for types of words and paras
0912 0 |
0913 0 LITERAL
0914 0     DELIMITED    = 0,
0915 0     NOT_DELIMITED = 1,
0916 0     WSPARA      = 0,
0917 0     EDTPARA     = 1;
```

G 4
15-Sep-1984 23:00:56
15-Sep-1984 22:43:32

VAX-11 Bliss-32 V4.0-742
_S255SDUA28:[EDT.SRC]EDT.REQ;1

```
: 0918 0 !+  
: 0919 0 ! Define the error codes.  
: 0920 0 !-  
: 0921 0 REQUIRE 'EDTSRC:ERRMSG.REQ';
```

R0922 0
R0923 0
R0924 0
R0925 0
R0926 0
R0927 0
R0928 0
R0929 0
R0930 0
R0931 0
R0932 0
R0933 0
R0934 0
R0935 0
R0936 0
R0937 0
R0938 0
R0939 0
R0940 0
R0941 0
R0942 0
R0943 0
R0944 0
R0945 0
R0946 0
R0947 0
R0948 0
R0949 0
R0950 0
R0951 0
R0952 0
R0953 0
R0954 0
R0955 0
R0956 0
R0957 0
R0958 0
R0959 0
R0960 0
R0961 0
R0962 0
R0963 0
R0964 0
R0965 0
R0966 0
R0967 0
R0968 0
R0969 0
R0970 0
R0971 0
R0972 0
R0973 0
R0974 0
R0975 0
R0976 0
R0977 0
R0978 0

```
*****  
*  
* 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.  
*  
*****
```

```
+ This file, ERRMSG.REQ, contains definitions of EDT's messages.  
Edit history:  
1-001 - Original, from EDT.REQ. JBS 20-Jul-1981  
1-002 - Add a short global name for PDP-11 MACRO modules. JBS 20-Jul-1981  
1-003 - Add a message for "work file failed to open". Module RSTIOI was  
using code 0 for this condition! JBS 20-Jul-1981  
1-004 - Change "" to "" in PDP-11 names. JBS 21-Jul-1981  
1-005 - Don't define the short global names, let EDT$MESSAGE do it. JBS 21-Jul-1981  
1-006 - Use an iterative macro. JBS 21-Jul-1981  
1-007 - Use the specified increment rather than 1 between messages. JBS 27-Jul-1981  
1-008 - Correct a couple of typos based on the MDL file. JBS 28-Jul-1981  
1-009 - Update a message based on the PDP-11 messages. JBS 28-Jul-1981  
1-010 - Change the form of the mnemonics to EDT$. Limit the names to nine characters,  
and let them be defined as globals on VAX-11. JBS 04-Aug-1981  
1-011 - Add new error msg INVSTR for bad string passed to set command. STS 20-Oct-1981  
1-012 - Make the INVSTR message more general so it can be used in more cases. JBS 22-Oct-1981  
1-013 - Correct a typo in NOFILSPC. SMB 03-Nov-1981  
1-014 - Revise messages FORHLPANO & TOEXITHLP. SMB 19-Nov-1981  
1-015 - Add a new message NOKEYHLP for no help on a key in change mode. SMB 20-Nov-1981  
1-016 - Add new messages for line number & sequence numbers out of range. SMB 3-Feb-1982  
1-017 - Revise existing error messages related to line numbers. SMB 05-Feb-1982  
1-018 - Add a new message for numeric value out of range. JBS 10-Feb-1982  
1-019 - Add a new message for ASCII character out of range. JBS 10-Feb-1982  
1-020 - Add a new message for internal EDT error. STS 19-Feb-1982  
1-021 - Add new message warning that DEFK is not allowed in Nokeypad mode. SMB 1-Mar-1982  
1-022 - Add message for passing back status. STS 09-Mar-1982  
1-023 - Add message for re-entry of EDT. STS 11-Mar-1982  
1-024 - Put the messages in alphabetical order. JBS 15-Mar-1982  
1-025 - Add a message for a non-standard input file. JBS 26-Mar-1982  
1-026 - Make the PDP-11 messages be status codes -- good ones are odd, bad are even. JBS 26-Mar-1982  
1-027 - Add messages about closing files. JBS 12-Apr-1982
```

I 4
15-Sep-1984 23:00:56
15-Sep-1984 22:44:02

VAX-11 Bliss-32 V4.0-742
_S255\$DUA28:[EDT.SRC]ERRMSG.REQ;1

Page 27
(1)

```
R0979 0 | 1-028 - Add second control C message. JBS 24-May-1982
R0980 0 | 1-029 - Add an error message for Help File initialization. SMB 28-May-1982
R0981 0 | 1-030 - Put the messages in the same order as the manual, to simplify
R0982 0 | verifying one against the other. JBS 09-Jun-1982
R0983 0 | 1-031 - Add 'Press return to continue'. JBS 17-Jun-1982
R0984 0 | 1-032 - Add 'Working'. JBS 18-Jun-1982
R0985 0 | 1-033 - Add a new select range error message. SMB 01-Jul-1982
R0986 0 | 1-034 - Do some miscellaneous improvements based on today's review. JBS 12-Jul-1982
R0987 0 | 1-035 - Fix duplicated mnemonic. JBS 13-Jul-1982
R0988 0 | 1-036 - Move ER_OUT and ER_INP to reflect the real meaning of the messages. SMB 13-Jul-1982
R0989 0 | 1-037 - Add an error message for no output file written and invalid
R0990 0 | input from terminal. STS 05-Aug-1982
R0991 0 | 1-038 - Take spaces out of the working message. SMB 18-Aug-1982
R0992 0 | 1-039 - Remove unused messages and add one new one. SMB 15-Dec-1982
R0993 0 | 1-040 - Add two new error messages for terminal opening. STS 15-Dec-1982
R0994 0 | 1-041 - Change the severity of terminal open errors to fatal. STS 16-Dec-1982
R0995 0 | 1-042 - Remove references to ASCII, since EDT uses the DEC Multinational
R0996 0 | character set. JBS 20-Jan-1983
R0997 0 | --
```

R0998 0
R0999 0
R1000 0
R1001 0
R1002 0
R1003 0
R1004 0
R1005 0
R1006 0
R1007 0
MR1008 0
MR1009 0
MR1010 0
MR1011 0
MR1012 0
MR1013 0
MR1014 0
MR1015 0
MR1016 0
MR1017 0
MR1018 0
MR1019 0
MR1020 0
MR1021 0
MR1022 0
MR1023 0
MR1024 0
MR1025 0
MR1026 0
MR1027 0
MR1028 0
MR1029 0
MR1030 0
MR1031 0
MR1032 0
MR1033 0
MR1034 0
MR1035 0
MR1036 0
MR1037 0
MR1038 0
MR1039 0
MR1040 0
MR1041 0
MR1042 0
MR1043 0
MR1044 0
MR1045 0
MR1046 0
MR1047 0
MR1048 0
MR1049 0
MR1050 0
MR1051 0
MR1052 0
MR1053 0
MR1054 0

+ Maintenance note: the messages should be kept in alphabetical order
by text, so that they can be matched against the manual.

+ Error messages: name, severity and text.

MACRO

ERROR_MESSAGES =

```
ERR (COLONREQ, W, ':' required',,,
ABOBYCC, W, 'Aborted by CTRL/C',,,
BOTOFBUF, W, 'Advance past bottom of buffer',,,
ASREQ, W, 'AS' required',,,
ATTCUTAPP, W, 'Attempt to CUT or APPEND to current buffer',,,
ATTPASCUR, W, 'Attempt to PASTE current buffer',,,
REENTRY, F, 'Attempt to re-enter EDT',,,
TOPOFBUF, W, 'Backup past top of buffer',,,
NOSETTRM, W, 'Cannot set terminal type from change mode',,,
CHGMODTER, W, 'Change mode can be entered only from a terminal',,,
COMBUFEXH, W, 'Command buffer exhausted',,,
COMEXHXL, W, 'Command buffer exhausted during XLATE command processing',,,
COMFILCLO, W, 'Command file could not be closed',,,
COMFILNOP, W, 'Command file could not be opened',,,
COMFILNEX, W, 'Command file does not exist',,,
CONCHKFLD, W, 'Consistency check failed, please check your file',,,
CLDNOTALN, W, 'Could not align tabs with cursor',,,
CTRC_IGN, W, 'CTRL/C ignored',,,
DSTMVOP, W, 'Destination for MOVE or COPY not found',,,
EDITORABO, F, 'Editor aborted',,,
ENTMUSTBE, W, 'Entity must be WORD, SENTENCE, PARAGRAPH or PAGE',,,
ERRCOMOPT, W, 'Error in command option',,,
ERRRANSPC, W, 'Error in range specification',,,
OPNINTRM, F, 'Error opening terminal for input',,,
OPNOUTTRM, F, 'Error opening terminal for output',,,
ERRINPFIL, W, 'Error reading from input file', ER_INP,
ERRINPTRM, W, 'Error reading from terminal',,,
ERROUTFIL, W, 'Error writing to output file', ER_OUT,
BADFILATR, W, 'File attributes error', ER_TYP,
FILNAM, W, 'File name:',
NOFILSPC, W, 'File specification required',,,
FORHLPANO, W, 'For help on any other keypad key, press the key',,,
HLPFILCLO, W, 'Help file could not be closed',,,
NOHLPVL, W, 'Help file could not be opened',,,
NOHLPINI, W, 'Help File Index could not be initialized',,,
INCFILCLO, W, 'Include file could not be closed',,,
INCFILOPN, W, 'Include file could not be opened',,,
INCFILNEX, W, 'Include file does not exist',,,
INPFILCLO, W, 'Input file could not be closed',,,
INPFILOPN, W, 'Input file could not be opened',,,
INPFILNEX, W, 'Input file does not exist',,,
NONSTDFIL, I, 'Input file does not have standard text file format', ER_NST,
INSMEM, W, 'Insufficient memory',,,
INTERERR, F, 'Internal software error - please submit an SPR',,,
INVBUFFNAM, W, 'Invalid buffer name',,,
```

```
MR1055 0 INVASCCHR, W, 'Invalid character code'..
MR1056 0 INVENT, W, 'Invalid entity'..
MR1057 0 INVOPTCOM, W, 'Invalid option for that command'..
MR1058 0 INVPARFOR, W, 'Invalid parameter for SET or SHOW'..
MR1059 0 INVSTR, W, 'Invalid string'..
MR1060 0 INVSUBCOM, W, 'Invalid subcommand'..
MR1061 0 INVVALSET, W, 'Invalid value in SET command'..
MR1062 0 IOERRWRK, F, 'I/O error on work file', ER_WF..
MR1063 0 JOUFILCLO, W, 'Journal file could not be closed'..
MR1064 0 NOJNLFIL, W, 'Journal file could not be opened'..
MR1065 0 BADDEFK, W, 'Keys cannot be defined in Nokeypad mode'..
MR1066 0 LINEXC255, W, 'Line exceeded 255 characters, truncated'..
MR1067 0 MACKEYREQ, W, 'MACRO or KEY required'..
MR1068 0 MAXINPLIN, F, 'Max input line of 2814749767 exceeded, file input terminated'..
MR1069 0 MAXLINNUM, F, 'Max line number exceeded; lines no longer ascending; resequence recommended'..
MR1070 0 MAXLINVAL, F, 'Max number of lines for this buffer exceeded'..
MR1071 0 NODEFN, W, 'No definition'..
MR1072 0 NOKEYHLP, W, 'No help available for that key'..
MR1073 0 BADRANGE, F, 'No more than 65535 lines can be processed in a single command'..
MR1074 0 NOFILWRT, W, 'No output file written'..
MR1075 0 NOSELNAN, W, 'No select range active'..
MR1076 0 NOSUCHLIN, W, 'No such line'..
MR1077 0 NOWENTDEF, W, 'Now enter the definition terminated by ENTER'..
MR1078 0 NUMVALILL, W, 'Numeric value illegal'..
MR1079 0 NUMVALREQ, W, 'Numeric value required'..
MR1080 0 NOORIGNUM, F, 'ORIGINAL line numbers no longer an EDT feature'..
MR1081 0 OUTFILCLO, W, 'Output file could not be closed'..
MR1082 0 OUTFILCRE, W, 'Output file could not be created'..
MR1083 0 PARENMISS, W, 'Parenthesis mismatch'..
MR1084 0 PARSTKOVF, W, 'Parsing stack overflow'..
MR1085 0 PLSANSYNQ, W, 'Please answer Y(es), N(o), Q(uit), or A(II)'..
MR1086 0 PASSTATUS, W, 'Pass bad status to caller'..
MR1087 0 PRERETCON, W, 'Press return to continue'..
MR1088 0 PRSKEYDEF, W, 'Press the key you wish to define'..
MR1089 0 PRIFILCLO, W, 'Print file could not be closed'..
MR1090 0 PRIFILCRE, W, 'Print file could not be created'..
MR1091 0 QUOSTRREQ, W, 'Quoted string required'..
MR1092 0 RANNONCON, W, 'Range for RESEQUENCE must be contiguous'..
MR1093 0 RANSPCSEQ, W, 'Range specified by /SEQUENCE would cause duplicate or non-sequential numbers'..
MR1094 0 RECTOOBIG, W, 'Record too big, truncated to 255 characters', ER_RTB..
MR1095 0 SUBSTRNUL, W, 'Search string cannot be null'..
MR1096 0 INVSAN, W, 'Select complete lines only'..
MR1097 0 SELALRACT, W, 'Select range is already active'..
MR1098 0 SEQINCROV, F, 'Sequence increment must be less than 65536'..
MR1099 0 SEQNUMOV, F, 'Sequence number must be less than 65536'..
MR1100 0 NONALPNUM, W, 'String delimiter must be non-alphanumeric'..
MR1101 0 STRNOTFND, W, 'String was not found'..
MR1102 0 KEYNOTDEF, W, 'That key is not definable'..
MR1103 0 TOEXITHLP, W, 'To exit from HELP, press the spacebar'..
MR1104 0 TORETKEY, W, 'To return to the keypad diagram, press the return key'..
MR1105 0 UNXCHRAFT, W, 'Unexpected characters after end of command'..
MR1106 0 UNRCOM, W, 'Unrecognized command'..
MR1107 0 UNRCOMOPT, W, 'Unrecognized command option'..
MR1108 0 WORFILCLO, W, 'Work file failed to close'..
MR1109 0 WORFILFAI, F, 'Work file failed to open', ER_WFO..
MR1110 0 WRKFILOVF, F, 'Work file overflow'..
MR1111 0 WORKING, F, 'Working'..
```

15⁴-Sep-1984 23:00:56
15-Sep-1984 22:44:02

VAX-11 Bliss-32 V4.0-742
_S255SDUA28:[EDT.SRC]ERRMSG.REQ;1

Page 30
(2)

: MR1112 0
: MR1113 0
: MR1114 0
: R1115 0 %:

WRIFILCLO, W, 'Write file could not be closed',
WRIFILCRE, W, 'Write file could not be created',
)

ED
VO

```
R1116 0
R1117 0
R1118 0
R1119 0
R1120 0
R1121 0
LR1122 0
UR1123 0
UR1124 0
UR1125 0
UR1126 0
UR1127 0
UR1128 0
UR1129 0
UR1130 0
UR1131 0
UR1132 0
UR1133 0
UR1134 0
UR1135 0
UR1136 0
UR1137 0
UR1138 0
UR1139 0
UR1140 0
UR1141 0
UR1142 0
UR1143 0
UR1144 0
UR1145 0
UR1146 0
UR1147 0
UR1148 0
UR1149 0
UR1150 0
UR1151 0
R1152 0
R1153 0
R1154 0
R1155 0
R1156 0
R1157 0
R1158 0
R1159 0
R1160 0
R1161 0
R1162 0
R1163 0
R1164 0
MR1165 0
MR1166 0
MR1167 0
MR1168 0
MR1169 0
MR1170 0
MR1171 0
MR1172 0

!+
! Define the base and offset for the message codes.
! The offset is used to distinguish EDT message codes from system-specific
! message codes. On VAX/VMS, the codes are defined by the MESSAGE compiler.
!-

%IF %BLISS (BLISS16)
%THEN

LITERAL
    W_BASE = 256,
    F_BASE = 256,
    I_BASE = 257,
    E_INC = 2;

!+
! Define the error codes.
!-

MACRO
    ERR [NAME, SEVERITY, TEXT, ENAME] =
        %NAME ('ERR ', NAME) =
            %NAME (SEVERITY, ' BASE') + (E_INC * ERROR_CODE)
    %;

COMPILETIME
    ERROR_CODE = 1;

LITERAL
    ERROR_MESSAGES;

UNDECLARE %QUOTE
    ERR;

%FI

!+
! The modules EDT$MESSAGE and EDT$MSGTXT use macro ERROR_MESSAGES to
! generate the text of each message.
!-

!+
! Define the MESSAGES macro, which defines EDT$mnemonic properly for
! either BLISS16 or BLISS32. On BLISS16 it is defined as a literal,
! equal to the ERR_mnemonic name. On BLISS32 it is defined as external.
!-

MACRO
    MESSAGES (MNEMONIC_LIST) =
        %IF %BLISS(BLISS16) %THEN
            MACRO MSG [MNEMONIC] =
                %NAME ('EDT$', MNEMONIC) = %NAME ('ERR_', MNEMONIC) %QUOTE %;
            LITERAL MSG (%REMOVE(MNEMONIC_LIST));
            UNDECLARE %QUOTE %QUOTE MSG;
        %ELSE
```

ED
Sy

ED
ED
ED
ED
ED

PS
--
\$\$

Ph
--
In
Co
Pa
Sy
Pa
Sy
Ps
Cr
As

Th
11
Th
68
1

Ma
--
_S

0

Th

MA

N 4
15-Sep-1984 23:00:56
15-Sep-1984 22:44:02

VAX-11 Bliss-32 V4.0-742
_S255\$DUA28:[EDT.SRC]ERRMSG.REQ;1

Page 32
(3)

: MR1173 0
: MR1174 0
: MR1175 0
: MR1176 0
: MR1177 0
: R1178 0
: R1179 0
: R1180 0
: R1181 0

MACRO MSG [MNEMONIC] =
%NAME('EDT\$', MNEMONIC) %QUOTE %;
EXTERNAL LITERAL MSG (%REMOVE(MNEMONIC_LIST));
UNDECLARE %QUOTE %QUOTE MSG;
%FI;
%;

!

End of file ERRMSG.REQ

```
1182 0
1183 0
1184 0
1185 0
1186 0
1187 0
1188 0
1189 0
1190 0
1191 0
1192 0
1193 0
1194 0
1195 0
1196 0
1197 0
1198 0
1199 0
1200 0
1201 0
1202 0
1203 0
1204 0
1205 0
1206 0
1207 0
1208 0
1209 0
1210 0

!+
Definition of the screen update data structure.

!-
This structure has an entry for each line which is represented on the screen.
In NOTRUNCATE mode, each record may occupy one or more screen lines.

!-
START_FIELDS(SCR_FIELDS)
  A_FIELD(SCR_PRV_LINE,%BPADDR),      ! Pointer to the previous line
  A_FIELD(SCR_NXT_LINE,%BPADDR),      ! Pointer to the next line
  A_FIELD(SCR_LINE_IDX,8),            ! The i'th screen line of this record
  A_FIELD(SCR_CHR_FROM,8),           ! Workfile char position from
  A_FIELD(SCR_CHR_TO,8),              ! Workfile char position to
  A_FIELD(SCR_EDIT_MINPOS,8),         ! Minimum position that has had an edit
  A_FIELD(SCR_EDIT_MAXPOS,8),         ! Maximum position that has had an edit
  A_FIELD(SCR_EDIT_FLAGS,8)          ! Modify, delete and insert flags
END_FIELDS

STRUC_SIZE(SCR_SIZE);

MACRO
  SCREEN_LINE = BLOCK[SCR_SIZE,BYTE] FIELD(SCR_FIELDS) %;

!+
These flags go in SCR_EDIT_FLAGS and are also used when calling EDT$$MRK_LNCHG.

!-
LITERAL
  SCR_EDIT_MODIFY = 1,                ! This line has been modified
  SCR_EDIT_INSLN = 2,                 ! This line has been inserted
  SCR_EDIT_DELLN = 4;                 ! This line has been deleted
```

```

: 1211 0
: 1212 0
: 1213 0
: 1214 0
: 1215 0
: 1216 0
: 1217 0
: 1218 0
: 1219 0
: 1220 0
: 1221 0
: 1222 0
U 1223 0
U 1224 0
U 1225 0
U 1226 0
U 1227 0
U 1228 0
U 1229 0
U 1230 0
: 1231 0

!+ This hack added to get around problem in CH$DIFF in BLISS16.
!-
%IF %BLISS(BLISS16) OR %BLISS(BLISS32) %THEN
    MACRO
        CH$PTR_GTR(P1,P2) = (P1) GTRA (P2) %,
        CH$PTR_GEQ(P1,P2) = (P1) GEQA (P2) %,
        CH$PTR_EQL(P1,P2) = (P1) EQLA (P2) %,
        CH$PTR_LEQ(P1,P2) = (P1) LEQA (P2) %,
        CH$PTR_LSS(P1,P2) = (P1) LSSA (P2) %,
        CH$PTR_NEQ(P1,P2) = (P1) NEQA (P2) %;
    %ELSE
        MACRO
            CH$PTR_GTR(P1,P2) = CH$DIFF(P1,P2) GTR 0 %,
            CH$PTR_GEQ(P1,P2) = CH$DIFF(P1,P2) GEQ 0 %,
            CH$PTR_EQL(P1,P2) = CH$DIFF(P1,P2) EQL 0 %,
            CH$PTR_LEQ(P1,P2) = CH$DIFF(P1,P2) LEQ 0 %,
            CH$PTR_LSS(P1,P2) = CH$DIFF(P1,P2) LSS 0 %,
            CH$PTR_NEQ(P1,P2) = CH$DIFF(P1,P2) NEQ 0 %;
        %FI

```

1232 0
1233 0
1234 0
1235 0
1236 0
1237 0
1238 0
1239 0
1240 0
1241 0
1242 0
1243 0
1244 0
1245 0
1246 0
1247 0
1248 0
1249 0
1250 0
1251 0
1252 0
1253 0
1254 0
1255 0
1256 0
1257 0
1258 0
1259 0
1260 0

!+ Define the entity types.
-

LITERAL

- ENT_K_CHAR = 1.
- ENT_K_WORD = 3.
- ENT_K_BW = 5.
- ENT_K_FW = 7.
- ENT_K_LINE = 9.
- ENT_K_BL = 11.
- ENT_K_NL = 13.
- ENT_K_VERT = 15.
- ENT_K_EL = 17.
- ENT_K_SEN = 19.
- ENT_K_BSEN = 21.
- ENT_K_ESEN = 23.
- ENT_K_PAR = 25.
- ENT_K_BPAR = 27.
- ENT_K_EPAR = 29.
- ENT_K_PAGE = 31.
- ENT_K_BPAGE = 33.
- ENT_K_EPAGE = 35.
- ENT_K_BR = 37.
- ENT_K_ER = 39.
- ENT_K_QUOTE = 41.
- ENT_K_SR = 43.
- LAST_R_ENT = 43.

1261 0
1262 0
1263 0
1264 0
1265 0
1266 0
1267 0
1268 0
1269 0
1270 0
1271 0
1272 0
1273 0
1274 0
1275 0
1276 0
1277 0
1278 0
1279 0
1280 0
1281 0
1282 0
1283 0
1284 0
1285 0
1286 0
1287 0
1288 0
1289 0
1290 0
1291 0
1292 0
1293 0
1294 0
1295 0
1296 0
1297 0
1298 0
1299 0
1300 0
1301 0
1302 0
1303 0
1304 0
1305 0
1306 0
1307 0
1308 0
1309 0
1310 0
1311 0
1312 0
1313 0
1314 0
1315 0
1316 0
1317 0

+ Define the verb numbers.
These are the codes used to represent the change mode subcommands.
The verbs from VERB_MOVE through VERB_APPEND require entities and their verb numbers must remain contiguous.
-

LITERAL

VERB_K_MOVE = 0.
VERB_K_DELETE = 1.
VERB_K_REPLACE = 2.
VERB_K_CHGC = 3.
VERB_K_CHGU = 4.
VERB_K_CHGL = 5.
VERB_K_SSEL = 6.
VERB_K_FILL = 7.
VERB_K_TADJ = 8.
VERB_K_CUT = 9.
VERB_K_APPEND = 10.

VERB_K_SEL = 11.

+ verbs verb_k_subs through verb_k_cc are special since they require variable length strings - keep them together with subs always first and cc last.
-

VERB_K_SUBS = 12.
VERB_K_PASTE = 13.
VERB_K_INSERT = 14.
VERB_K_XLATE = 15.
VERB_K_CC = 16.
VERB_K_EXIT = 17.
VERB_K_SN = 18.
VERB_K_UNDC = 19.
VERB_K_UNDW = 20.
VERB_K_UNDL = 21.
VERB_K_ADV = 22.
VERB_K_BACK = 23.
VERB_K_REF = 24.
VERB_K_TOP = 25.
VERB_K_HELP = 26.
VERB_K_ASC = 27.
VERB_K_QUIT = 28.
VERB_K_SHL = 29.
VERB_K_SHR = 30.
VERB_K_TAB = 31.
VERB_K_TC = 32.
VERB_K_TD = 33.
VERB_K_TI = 34.
VERB_K_EXT = 35.
VERB_K_KS = 36.
VERB_K_DEFK = 37.
VERB_K_BELL = 38.
VERB_K_DATE = 39.

```
: 1318 0 VERB_K_DUPC = 40,  
: 1319 0 VERB_K_DLWC = 41,  
: 1320 0 VERB_K_DMOV = 42,  
: 1321 0 VERB_K_DESEL = 43,  
: 1322 0 VERB_K_TGSEL = 44,  
: 1323 0 VERB_K_CLSS = 45,  
: 1324 0 LAST_K_VERB = 45;  
: 1325 0  
: 1326 0  
: 1327 0 ! Changecase types.  
: 1328 0 !  
: 1329 0  
: 1330 0 LITERAL  
: 1331 0 CASE_K_CHGC = 1, ! Invert case, corresponds to VERB_K_CHGC  
: 1332 0 CASE_K_CHGU = 2, ! Upper case, corresponds to VERB_K_CHGU  
: 1333 0 CASE_K_CHGL = 3, ! Lower case, corresponds to VERB_K_CHGL
```

1334 0
1335 0
1336 0
1337 0
1338 0
1339 0
1340 0
1341 0
1342 0
1343 0
1344 0
1345 0
1346 0
1347 0
1348 0
1349 0
1350 0
1351 0
1352 0
1353 0
1354 0
1355 0
1356 0
1357 0
1358 0
1359 0
1360 0
1361 0
1362 0
1363 0
1364 0
1365 0
1366 0
1367 0
1368 0
1369 0
1370 0
1371 0
1372 0
1373 0
1374 0
1375 0
1376 0
1377 0
1378 0
1379 0

```
!+
!-
PARSER OP-CODE DEFINITIONS

The following are the op-codes accepted by the parser driver.

LITERAL
OPC_ABORT      = 0.      ! Abort the parse
OPC_ACTION     = 1.      ! Perform action routine
OPC_CALL       = 2.      ! Call sub-table
OPC_RETURN     = 3.      ! End of table or sub-table (return)
OPC_GOTO       = 4.      ! Unconditional goto
OPC_OPTION     = 5.      ! Optional phrase check
OPC_REQUIRE    = 6.      ! Require a specific token
OPC_SELECT     = 7.      ! select one of several options

OP_ABORT      = 0.      ! now the bit values
OP_ACTION     = 32.
OP_CALL       = 64.
OP_RETURN     = 96.
OP_GOTO       = 128.
OP_OPTION     = 160.
OP_REQUIRE    = 192.
OP_SELECT     = 224.
```

```
!+
!-
Token class definitions

LITERAL
CL_NAME       = 0.      ! name class
CL_NUMBER     = 1.      ! the number class
CL_SPECIAL    = 2.      ! the special character class
CL_STRING     = 3.      ! The quoted string class
```

```
!+
!-
Parser token handling and matching macros

MACRO
PAR_MIN_LENGTH = 0,0,3,0 %
PAR_MAX_LENGTH = 0,4,4,0 %
PAR_OPT_PERCENT = 0,3,1,0 %
PAR_SYMBOL     = 1,0,0,0 %;
```

```
1380 0
1381 00
1382 00
1383 00
1384 00
1385 00
1386 00
1387 00
M 1388 00
1389 00
1390 00
U 1391 00
U 1392 00
U 1393 00
U 1394 00
U 1395 00
U 1396 00
U 1397 00
U 1398 00
U 1399 00
U 1400 00
U 1401 00
U 1402 00
U 1403 00
U 1404 00
U 1405 00
U 1406 00
U 1407 00
U 1408 00
U 1409 00
U 1410 00
U 1411 00
1412 00
1413 00
1414 00
1415 00
1416 00
1417 00
1418 00
1419 00
1420 00
1421 00
1422 00
1423 00
1424 00
1425 00
1426 00
M 1427 00
M 1428 00
M 1429 00
M 1430 00
M 1431 00
M 1432 00
M 1433 00
M 1434 00
M 1435 00
1436 0
```

!+
Miscellaneous definitions
-

%IF %BLISS(BLISS32) %THEN

MACRO STRING_DESC(DESC,LEN,ADDR) =
BEGIN EXTERNAL ROUTINE STR\$COPY_R; STR\$COPY_R(DESC,LEN,ADDR) END %;

%ELSE
!+
These DSC\$ macros are defined as system symbols on VAX/VMS. They are
fields in a string descriptor. To get the effect of a string descriptor
on the 11's, we will pass a 4 word field with the following macros defining
the pointer to the string address and the field of the string length.
-

MACRO
DSC\$A_POINTER = 4,0,16,0%;
DSC\$W_LENGTH = 0,0,16,0%;

MACRO STRING_DESC (DESC, LEN, ADDR) =
BEGIN
MAP
DESC: BLOCK[8,BYTE];
DESC[DSC\$A_POINTER] = ADDR;
DESC[DSC\$W_LENGTH] = .LEN;
END %;

%FI

LITERAL
NO_UPDATE = 256, ! Indicating no update of current line needed
NO_REFRESH = 100, ! Indicating no refresh of screen needed
MESSAGE_LINE= 22, ! Line on which messages are displayed
COMMAND_LINE= 23, ! Line on which command prompts are displayed
DIR_FORWARD = 1, ! Forward direction.
DIR_BACKWARD= 0; ! Backward direction.

!+
Definition of the ASSERT macro. This macro calls EDT\$\$INTER_ERR if the
condition is not true.
-

MACRO ASSERT (CONDITION) =
BEGIN
IF (NOT (CONDITION))
THEN
BEGIN
EXTERNAL ROUTINE EDT\$\$INTER_ERR : NOVALUE;
EDT\$\$INTER_ERR ();
END;
END
%;

```
: 1437 0 !+ Symbols used in control C journaling.
: 1438 0 !-
: 1439 0 LITERAL
: 1440 0 CC_REC_SIZE = 6, ! Size of a control C record
: 1441 0 JOD_REC_ESC = %X'FF', ! First (escape) byte of a non-text record in the journal file
: 1442 0 CC_REC_FLAG = 1, ! Second byte: control C record
: 1443 0 CC_CTR_MAX = 30000; ! Maximum counter value in control C handling
: 1444 0
: 1445 0 !+ Symbol used in the formatter
: 1446 0 !-
: 1447 0 %IF %BLISS(BLISS32) %THEN
: 1448 0 LITERAL
: 1449 0 EDT$K_FMT_BUFLen = 512; ! Length of the format buffer
: 1450 0
: 1451 0 %ELSE
: 1452 0 LITERAL
: 1453 0 EDT$K_FMT_BUFLen = 136; ! Length of the format buffer
: 1454 0
: 1455 0 %FI
: 1456 0
: 1457 0
: 1458 0
: 1459 0 ! End of file EDT.REQ
```

COMMAND QUALIFIERS

BLISS/LIBRARY=EDTSRC:EDT/LIST=LIS\$:/SOURCE=REQUIRE SRC\$:EDT.REQ

: Run Time: 00:12.2
: Elapsed Time: 00:23.8
: Lines/CPU Min: 7193
: Lexemes/CPU-Min: 36428
: Memory Used: 104 pages
: Library Precompilation Complete

EXTEND
LIS

FDEC
LIS

FILL
LIS

FINDPARA
LIS

FCRIF
LIS

EDT
LIS

EXEC
LIS

EXECNOO
LIS

FILEIO
LIS

EDTVECTOR
LIS

FINDKEY
LIS

FCOINC
LIS

FINAL
LIS

FIOHDLR
LIS

DEFKEY
LIS

ERRMSG
LIS

FCHAR
LIS