

DDDDDDDDDDDD		CCCCCCCCCCCC	LLL
DDDDDDDDDDDD		CCCCCCCCCCCC	LLL
DDDDDDDDDDDD		CCCCCCCCCCCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDD	DDD	CCC	LLL
DDDDDDDDDDDD		CCCCCCCCCCCC	LLLLLLLLLLLLLLLL
DDDDDDDDDDDD		CCCCCCCCCCCC	LLLLLLLLLLLLLLLL
DDDDDDDDDDDD		CCCCCCCCCCCC	LLLLLLLLLLLLLLLL

```

EEEEEEEEEE XX XX PPPPPPPP RRRRRRRR EEEEEEEEE SSSSSSSS SSSSSSSS
EEEEEEEEEE XX XX PPPPPPPP RRRRRRRR EEEEEEEEE SSSSSSSS SSSSSSSS
EE XX XX PP PP RR RR EE SS SS
EE XX XX PP PP RR RR EE SS SS
EE XX XX PP PP RR RR EE SS SS
EEEEEEEE XX XX PPPPPPPP RRRRRRRR EEEEEEEEE SSSSSSS SSSSSSS
EEEEEEEE XX XX PPPPPPPP RRRRRRRR EEEEEEEEE SSSSSSS SSSSSSS
EE XX XX PP PP RR RR EE SS SS
EE XX XX PP PP RR RR EE SS SS
EE XX XX PP PP RR RR EE SS SS
EEEEEEEEEE XX XX PP PP RR RR EEEEEEEEEEE SSSSSSSS SSSSSSSS
EEEEEEEEEE XX XX PP PP RR RR EEEEEEEEEEE SSSSSSSS SSSSSSSS

```

```

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
LLLLLLLLLLL IIIIII SSSSSSSS
LLLLLLLLLLL IIIIII SSSSSSSS

```

(3)	169	DUMMY LIB\$SCOPY_DXD ROUTINE
(4)	174	EVALUATE BINARY-EXPRESSION
(5)	198	CONVERT STRING OPERAND TO BINARY
(6)	245	EXPRESSION ANALYSIS
(7)	481	SEARCH OPERATOR TABLE
(8)	519	PARSE_OPERAND - PARSE_OPERAND_TOKEN
(10)	694	FETCH_NEXT_OPERAND_FROM_TRIAD_STACK
(11)	704	GET_OPERAND_DESCRIPTOR
(12)	718	CONVERT_STRING_PARAMETERS_TO_STRING_OPERAND
(13)	736	CONVERT_OPERAND_PARAMETERS_TO_NUMERIC_VALUE
(14)	769	DISPATCH_STRING_OPERATION_FUNCTION
(15)	777	STRING_CONCATENATION_OPERATOR
(16)	809	STRING_REDUCTION_OPERATOR
(17)	846	DISPATCH_BINARY/LOGICAL_OPERATOR

```
0000 1      .TITLE EXPRESS - EXPRESSION ANALYSIS
0000 2      .IDENT 'V04-000'
0000 3
0000 4
0000 5 :*****
0000 6 :*
0000 7 :*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 8 :*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 9 :*  ALL RIGHTS RESERVED.
0000 10 :*
0000 11 :*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 12 :*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 13 :*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 14 :*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 15 :*  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 16 :*  TRANSFERRED.
0000 17 :*
0000 18 :*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 19 :*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 20 :*  CORPORATION.
0000 21 :*
0000 22 :*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 23 :*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 24 :*
0000 25 :*
0000 26 :*****
0000 27
0000 28 : EXPRESSION ANALYSIS
0000 29
0000 30 : D. N. CUTLER 9-MAY-77
0000 31
0000 32 : MODIFIED BY:
0000 33
0000 34 :      V03-003 PCG0007      Peter George      27-Jul-1983
0000 35 :      Add dummy LIB$SCOPY_DXDX routine.
0000 36
0000 37 :      V03-002 PCG0006      Peter George      12-Jul-1983
0000 38 :      Fix bug in buffer overflow calculation.
0000 39
0000 40 :      V03-001 PCG0005      Peter George      18-Nov-1982
0000 41 :      Upcase symbol names when evaluating F$VERIFY arguments
0000 42 :      in a comment.
0000 43
0000 44 :---
```

```

0000 46 :
0000 47 : MACRO LIBRARY CALLS
0000 48 :
0000 49 :
0000 50          PRCDEF          ;DEFINE PROCESS WORK AREA
0000 51          WRKDEF          ;DEFINE COMMAND WORK AREA
0000 52          $CLIMSGDEF      ;DEFINE ERROR/STATUS CODES
0000 53 :
0000 54 :
0000 55 : LOCAL MACROS
0000 56 :
0000 57 : GENERATE OPERATOR/OPERAND TABLE AND SYMBOL NAMES
0000 58 :
0000 59 :
0000 60          .MACRO GENOP NAME,INDEX,TYPE,PREC,CHAR,?L1,?L2,?L3,?L4
0000 61 OPI_K_'NAME=INDEX
0000 62          .IF IDN <TYPE>,<OPERATOR>
0000 63 OPP_K_'NAME=PREC
0000 64          .IF DIF <NAME>,<SOS>
0000 65          .IF DIF <NAME>,<EOS>
0000 66          .IF DIF <NAME>,<STORE>
0000 67 L1:      .BYTE          L4-L1
0000 68          .BYTE          L3-L2
0000 69          .IF NB <CHAR>
0000 70 L2:      .ASCII  \CHAR\
0000 71          .IFF
0000 72 L2:      .ASCII  /NAME/
0000 73          .ENDC
0000 74 L3:      .BYTE          OPP_K_'NAME
0000 75 L4:      .BYTE          OPI_K_'NAME
0000 76          .ENDC
0000 77          .ENDC
0000 78          .ENDC
0000 79          .ENDC
0000 80          .ENDM  GENOP
0000 81 :
0000 82 :
0000 83 : LOCAL SYMBOLS
0000 84 :
0000 85 : DEFINE STACK ITEM OFFSETS
0000 86 :
0000 87 :
0000 88          $DEFINI STK
0000 89 :
0000 90 $DEF      STK_W_TYPE      .BLKW  1      ;STACK ITEM TYPE
0002 91 $DEF      STK_W_PREC     .BLKW  1      ;OPERATOR PRECEDENCE
0002 92 $DEF      STK_W_SIZE     .BLKW  1      ;SIZE OF OPERAND VALUE IN BYTES
0004 93 $DEF      STK_L_ADDR     .BLKL  1      ;ADDRESS OF OPERAND VALUE
0008 94 $DEF      STK_K_LENGTH   .BLKW  1      ;LENGTH OF STACK ITEM
0008 95 :
0008 96          $DEFEND STK
0000 97 :
0000 98 :
0000 99 : LOCAL DATA
0000 100 :
0000 101 : OPERATOR TABLE
0000 102 :

```

```

0000 103
00000000 104 .PSECT DCL$ZCODE, BYTE, RD, NOWRT
0000 105
0000 106 OPTAB: GENOP SOS, 30, OPERATOR, 0 ; START OF STATEMENT
0000 107 GENOP EOS, 32, OPERATOR, 1 ; END OF STATEMENT
0000 108 GENOP LPAREN, 34, OPERATOR, 2, <(> ; LEFT PARENTHESIS
0005 109 GENOP RPAREN, 36, OPERATOR, 3, <)> ; RIGHT PARENTHESIS
000A 110 GENOP STORE, 38, OPERATOR, 0 ; STORE RESULT
000A 111 GENOP AND, 0, OPERATOR, 5 ; BOOLEAN AND
0011 112 GENOP NOT, 2, OPERATOR, 6 ; BOOLEAN NOT
0018 113 GENOP OR, 4, OPERATOR, 4 ; BOOLEAN OR
001E 114 GENOP ADD, 6, OPERATOR, 8, <+> ; INTEGER ADD
0023 115 GENOP SUB, 8, OPERATOR, 8, <-> ; INTEGER SUBTRACT
0028 116 GENOP MUL, 10, OPERATOR, 9, <*> ; INTEGER MULTIPLY
002D 117 GENOP DIV, 12, OPERATOR, 9, </> ; INTEGER DIVIDE
0032 118 GENOP NEG, 14, SPECIAL ; INTEGER NEGATE
0032 119 GENOP POS, 16, SPECIAL ; INTEGER NOOP
0000000A 0032 120 OPP_K_NEG=OPP_K_DIV+1 ; ONE GREATER THAN HIGEST ARITHMETIC
0000000A 0032 121 OPP_K_POS=OPP_K_DIV+1 ; ONE GREATER THAN HIGEST ARITHMETIC
0032 122 GENOP EQ, 18, OPERATOR, 7 ; ARITHMETIC EQUAL
0038 123 GENOP GE, 20, OPERATOR, 7 ; ARITHMETIC GREATER OR EQUAL
003E 124 GENOP GT, 22, OPERATOR, 7 ; ARITHMETIC GREATER
0044 125 GENOP LE, 24, OPERATOR, 7 ; ARITHMETIC LESS OR EQUAL
004A 126 GENOP LT, 26, OPERATOR, 7 ; ARITHMETIC LESS
0050 127 GENOP NE, 28, OPERATOR, 7 ; ARITHMETIC NOT EQUAL
0056 128 GENOP EQS, 40, OPERATOR, 7 ; STRING EQUAL
005D 129 GENOP GES, 42, OPERATOR, 7 ; STRING GREATER OR EQUAL
0064 130 GENOP GTS, 44, OPERATOR, 7 ; STRING GREATER
006B 131 GENOP LES, 46, OPERATOR, 7 ; STRING LESS OR EQUAL
0072 132 GENOP LTS, 48, OPERATOR, 7 ; STRING LESS
0079 133 GENOP NES, 50, OPERATOR, 7 ; STRING NOT EQUAL
00000034 0080 134 OPI_K_OPERAND = 52 ; LOWEST OPERAND INDEX
0080 135 GENOP STRING, 52, OPERAND ; UNEVALUATED CHARACTER STRING OPERAND
0080 136 GENOP STACK, 54, OPERAND ; EVALUATED ON-STACK OPERAND
00 0080 137 .BYTE 0 ; TABLE TERMINATOR BYTE
0081 138
0081 139 ;
0081 140 ; SIZE OF EXPRESSION ANALYSIS STACK
0081 141 ;
0081 142
00000084 0081 143 PARSESTKSIZ = 132 ; SIZE OF PARSE STACK
0000016C 0081 144 TRIADSTKSIZ = 364 ; SIZE OF TRIAD STACK
0081 145
0081 146 ;
0081 147 ; LOCAL VARIABLES ON STACK
0081 148 ;
0081 149
0081 150 $DEFINI
00000001 0000 151 NESTLVL: .BLKB 1 ; CURRENT LEVEL OF NESTING
00000002 0001 152 REQMODE: .BLKB 1 ; REQUIRED EXPRESSION MODE
0002 153 ; (0=CAN BE ANY MODE)
00000003 0002 154 CURMODE: .BLKB 1 ; CURRENT EXPRESSION MODE
00000004 0003 155 .BLKB 1
0000000C 0004 156 RESULT: .BLKB 1 ; EXPRESSION RESULT
000C 157 LOCALSIZ:
000C 158 $DEFEND
0081 159

```

```
0081 160 :  
0081 161 : 'TRUE' AND 'FALSE' VALUES  
0081 162 :  
0081 163 :  
45 45 55 52 54 0081 164 TRUE: .ASCII 'TRUE' :  
53 4C 41 46 0085 165 FALSE: .ASCII 'FALSE' :  
79 59 74 54 008A 166 TRUSYM: .ASCII 'TtYy' :  
008E 167 : NOTE-ORDER IS ASSUMED ELSEWHERE
```

EXPRESS  
V04-000

- EXPRESSION ANALYSIS  
DUMMY LIB\$SCOPE\_DXXD ROUTINE

M 1

15-SEP-1984 23:46:42  
4-SEP-1984 23:40:31

VAX/VMS Macro V04-00  
[DCL.SRC]EXPRESS.MAR;1

Page 5  
(3)

	008E	169	.SBTTL	DUMMY LIB\$SCOPE_DXXD ROUTINE
	008E	170		
0000	008E	171	.ENTRY	LIB\$SCOPE_DXXD.0
04	0090	172	RET	



```

0091 174      .SBTTL  EVALUATE BINARY EXPRESSION
0091 175      :+
0091 176      : DCL$BINEXPR - EVALUATE BINARY EXPRESSION
0091 177      :
0091 178      : THIS ROUTINE IS CALLED TO ANALYZE AN EXPRESSION AND RETURN AN ARITHMETIC RESULT.
0091 179      :
0091 180      : INPUTS:
0091 181      :
0091 182      :     R10 = BASE ADDRESS OF COMMAND WORK AREA.
0091 183      :     R11 = BASE ADDRESS OF PROCESS WORK AREA.
0091 184      :
0091 185      : OUTPUTS:
0091 186      :
0091 187      :     R0 = STATUS
0091 188      :     R1 = BINARY LONGWORD VALUE
0091 189      :
0091 190      :     R2,R3 DESTROYED.
0091 191      : -
0091 192      :
0091 193      DCL$BINEXPR::
01 2D 10 0091 194      BSBB      DCL$EXPRESS      :EVALUATE EXPRESSION
01 50 E8 0093 195      BLBS      R0,DCL$CVT_BINARY  :IF OK, CONVERT RESULT TO BINARY
0096 05 0096 196      RSB

```

```

0097 198 .SBTTL CONVERT STRING OPERAND TO BINARY
0097 199 :+
0097 200 : DCL$CVT_BINARY - CONVERT STRING OPERAND TO BINARY
0097 201 :
0097 202 : THIS ROUTINE IS CALLED TO CONVERT A EXPRESSION RESULT TO BINARY.
0097 203 : IF THE RESULT IS ALREADY BINARY, THEN THAT VALUE IS RETURNED.
0097 204 :
0097 205 : INPUTS:
0097 206 :
0097 207 : R10 = BASE ADDRESS OF COMMAND WORK AREA.
0097 208 : R11 = BASE ADDRESS OF PROCESS WORK AREA.
0097 209 :
0097 210 : R1/R2 = QUADWORD DESCRIBING EXPRESSION VALUE:
0097 211 : IF R2 NONZERO, R1/R2 ARE A STRING DESCRIPTOR
0097 212 : IF R2 ZERO, R1 IS A BINARY LONGWORD VALUE
0097 213 :
0097 214 : OUTPUTS:
0097 215 :
0097 216 : R0 = STATUS
0097 217 : R1 = BINARY LONGWORD VALUE
0097 218 : R2 = 0 (TO INDICATE RESULT IS BINARY)
0097 219 :
0097 220 : R3 DESTROYED.
0097 221 :-
0097 222 :
0097 223 DCL$CVT_BINARY::
51 52 D5 0097 224 TSTL R2 ;STRING OR BINARY VALUE?
1F 13 0099 225 BEQL 80$ ;OK IF BINARY VALUE
52 DD 009B 226 PUSHL R2 ;SAVE STARTING ADDRESS OF STRING
51 52 51 7D 009D 227 MOVQ R1,R2 ;COPY STRING DESCRIPTOR
00AE CB 9A 00A0 228 MOVZBL PRC B DEFRADIX(R11),R1 ;SET RADIX FOR CONVERSION
FF58 30 00A5 229 BSBW DCL$CNVASCBIN ;CONVERT TO BINARY
04 BA 00A8 230 POPR #*M<R2> ;RESTORE STARTING ADDRESS OF STRING
0E 13 00AA 231 BEQL 80$ ;IF SUCCESSFUL, EXIT WITH R1
00AC 232 :
00AC 233 : INVALID CHARACTER IN STRING-CHECK FOR BEGINNING LETTER "tYy"
00AC 234 : IF "T", "t", "Y", OR "y", THEN RETURN 1, ELSE RETURN 0.
00AC 235 :
D9 AF 04 62 3A 00AC 236 LOCC (R2),#4,TRUSYM ;CHECK FOR TRUE CHARACTERS
05 13 00B1 237 BEQL 70$ ;BRANCH IF NONE FOUND
51 01 D0 00B3 238 MOVL #1,R1 ;SET RESULT TO TRUE
02 11 00B6 239 BRB 80$ ;SKIP
51 D4 00B8 240 70$: CLRL R1 ;SET RESULT TO FALSE
50 01 D0 00BA 241 80$: MOVL #1,R0 ;SUCCESSFUL
52 D4 00BD 242 CLRL R2 ;INDICATE RESULT IS BINARY
05 00BF 243 RSB

```

```

00C0 245 .SBTTL EXPRESSION ANALYSIS
00C0 246 :+
00C0 247 : DCL$EXPRESS - EXPRESSION ANALYSIS
00C0 248 :
00C0 249 : THIS ROUTINE IS CALLED TO ANALYZE AN EXPRESSION AND RETURN A BINARY
00C0 250 : OR CHARACTER STRING RESULTANT VALUE.
00C0 251 :
00C0 252 : INPUTS:
00C0 253 :
00C0 254 : R10 = BASE ADDRESS OF COMMAND WORK AREA.
00C0 255 : R11 = BASE ADDRESS OF PROCESS WORK AREA.
00C0 256 :
00C0 257 : OUTPUTS:
00C0 258 :
00C0 259 : R0 = STATUS
00C0 260 : R1/R2 = QUADWORD DESCRIBING EXPRESSION VALUE:
00C0 261 : IF R2 NONZERO, R1/R2 ARE A STRING DESCRIPTOR
00C0 262 : IF R2 ZERO, R1 IS A BINARY LONGWORD VALUE
00C0 263 :
00C0 264 : R3 DESTROYED.
00C0 265 :-
00C0 266
00C0 267 DCL$EXPRESS:: :ANALYZE EXPRESSION
51 01 90 00C0 268 MOV B #PRC_K_DEC,R1 :SET TO DEFAULT DECIMAL RADIX
13F0 8F BB 00C3 269 DCL$EXPRADIX:: :ALTERNATE ENTRY-RADIX SET EXTERNALLY
F486 CA DD 00C7 270 PUSH R #^M<R4,R5,R6,R7,R8,R9,AP> :SAVE REGISTERS
00AE CB 51 90 00CB 271 PUSH L WRK_L_EXPANDPTR(R10) :SAVE CURRENT PARSE POSITION
FO AA 20 A8 00D0 272 MOV B R1,PRC_B_DEFRADIX(R11) :SET THE RADIX FOR LATER
5E 0C C2 00D4 273 BIS W #WRK_M_STAR,WRK_W_FLAGS(R10) :SET ASTERISK TERMINATOR FLAG
57 5E D0 00D7 274
01 A7 53 90 00DC 275 SUB L #LOCALSIZ,SP :ALLOCATE SCRATCH SPACE ON STACK
58 5E D0 00E0 276 MOV L SP,R7 :POINT TO SCRATCH SPACE
FE10 CE 9E 00E3 277 CLR B NESTLVL(R7) :INITIALIZE PARENTHESIS NESTING LEVEL
78 1E D0 00EB 278 MOV B R3,REQMODE(R7) :SET EXPECTED EXPRESSION MODE
78 68 7D 00EE 279
SE 58 5E D0 00E0 280 MOV L SP,R8 :SET ADDRESS OF PARSE STACK
FE10 CE 9E 00E3 281 MOV AB -<TRIADSTKSIZ+PARSESTKSIZ>(SP),SP :ALLOCATE SPACE FOR STACKS
59 5E D0 00EB 282 MOV L SP,R9 :SET BASE ADDRESS OF TRIAD STACK
78 1E D0 00EB 283 MOV L #<OPP_K_SOS>@16+OPI_K_SOS,-(R8) :INITIALIZE PARSE STACK
78 68 7D 00EE 284 MOV Q (R8),-(R8) :DUPLICATE FIRST ITEM FOR ERROR CHECK
00F1 285
00F1 286 :
00F1 287 : PARSE NEXT ITEM
00F1 288 :
00F1 289
50 FFOC' 30 00F1 290 10$: BSBW DCL$SETCHAR :PEEK AT NEXT CHARACTER IN INPUT BUFFER
20 20 91 00F4 291 CMP B #^A/ /,R0 :BLANK?
03 12 00F7 292 BNE Q 20$ :IF NEQ NO
FF04' 30 00F9 293 BSBW DCL$MOVCHAR :MOVE CHARACTER TO COMMAND BUFFER
FF01' 30 00FC 294 20$: BSBW DCL$MARK :MARK CURRENT PARSE POSITION
FEFE' 30 00FF 295 BSBW DCL$SETCHAR :PEEK AT NEXT CHARACTER IN INPUT BUFFER
50 2E 91 0102 296 CMP B #^A/ ./,R0 :OPERATOR?
21 13 0105 297 BEQ L 25$ :IF NEQ NO
FEF6' 30 0107 298 BSBW DCL$GETOKEN :GET NEXT TOKEN FROM INPUT BUFFER
30 13 010A 299 BEQ L 40$ :IF EQL NONE
34 68 B1 010C 300 CMP W STK_W_TYPE(R8),#OPI_K_OPERAND :TOS AN OPERAND?
OE 19 010F 301 BLS S 80$ :BR IF OPERATOR IS TOP

```

```

51 04 D1 0111 302      CMPL  #4,R1      ;POSSIBLY 'THEN' KEY WORD?
      09 12 0114 303      BNEQ  80$      ;IF NEQ NO
62 4E454854 8F D1 0116 304      CMPL  #^A/THEN/, (R2) ;'THEN' KEYWORD?
      3C 13 0115 305      BEQL  130$     ;IF SO, TERMINATE EXPRESSION PARSE
      011F 306
      011F 307
      011F 308 : PARSE OPERAND TOKEN
      011F 309
      011F 310
      015A 30 011F 311 80$:  BSBW  PARSE_OPERAND ;PARSE THE OPERAND
      0122 312 ;(RESULTS MAY BE PLACED ON STACK)
      14 50 E9 0122 313      BLBC  RO,35$ ;EXIT IF ERROR DETECTED
      00A6 31 0125 314      BRW   170$ ;STACK THE OPERAND PARAMETERS
      0128 315
      0128 316 :
      0128 317 : PARSE .XX. OPERATORS
      0128 318
      0128 319
      50 FED5' 30 0128 320 25$:  BSBW  DCL$MOVBTKN ;MOVE TERMINATOR AND GET NEXT BLANK TOKEN
      2E 91 012B 321      CMPB  #^A/./,RO ;TERMINATOR PERIOD?
      03 12 012E 322      BNEQ  30$ ;IF NEQ NO
      FECD' 30 0130 323      BSBW  DCL$MOVCHAR ;MOVE TERMINATOR TO EXPANSION BUFFER
      0114 30 0133 324 30$:  BSBW  FIND_OPERATOR ;SEARCH OPERATOR TABLE
      2A 50 E8 0136 325      BLBS  RO,50$ ;IF FOUND, PROCESS OPERATOR
      02AA 31 0139 326 35$:  BRW   EXIT ;EXIT WITH ERROR
      013C 327
      013C 328 :
      013C 329 : PARSE OPERATOR CHARACTER
      013C 330
      013C 331 : TERMINATE EXPRESSION IF END-OF-LINE OR RIGHT PAREN W/O CORRESP. LEFT
      013C 332
      013C 333
      52 F486 CA D0 013C 334 40$:  MOVL  WRK_L_EXPANDPTR(R10),R2 ;SET ADDRESS OF START OF SYMBOL
      29 50 91 0141 335      CMPB  RO,#^A')' ;RIGHT PAREN?
      04 12 0144 336      BNEQ  45$ ;BRANCH IF NOT
      67 95 0146 337      TSTB  NESTLVL(R7) ;IS THERE A CORRESPONDING LEFT PAREN?
      11 15 0148 338      BLEQ  130$ ;IF NO LEFT PAREN, TERMINATE EXPRESSION
      FEB3' 30 014A 339 45$:  BSBW  DCL$MOVCHAR ;MOVE TERMINATOR TO EXPANSION BUFFER
      0C 13 014D 340      BEQL  130$ ;IF EQL END OF LINE
      51 01 D0 014F 341      MOVL  #1,R1 ;SET LENGTH OF STRING
      00F5 30 0152 342      BSBW  FIND_OPERATOR ;SEARCH OPERATOR TABLE
      0B 50 E8 0155 343      BLBS  RO,50$ ;IF FOUND, PROCESS THE OPERATOR
      FEAS' 30 0158 344      BSBW  DCL$BACKUPMOVE ;BACKUP MOST RECENT MOVCHAR
      015B 345
      015B 346 :
      015B 347 : MARK END OF EXPRESSION
      015B 348
      015B 349
      51 01 9A 015B 350 130$:  MOVZBL #OPP_K_EOS,R1 ;SET END OF STATEMENT PRECEDENCE VALUE
      53 20 9A 015E 351      MOVZBL #OPI_K_EOS,R3 ;SET END OF STATEMENT OPERATOR INDEX
      36 11 0161 352      BRB   140$ ;FORCE EVERYTHING TO TRIAD STACK
      0163 353
      0163 354 :
      0163 355 : PROCESS OPERATOR
      0163 356
      0163 357 : IF LEFT PARENTHESIS, INCREMENT THE NESTING LEVEL AND MAKE SURE
      0163 358 : AN OPERATOR PRECEEDS IT.

```

```

0163 359 ;
0163 360 ;
53 22 D1 0163 361 50$:  CMPL  #OPI_K_LPAREN,R3      ;LEFT PARENTHESIS?
    OA 12 0166 362      BNEQ  53$      ;IF NEQ NO
    67 96 0168 363      INCB  NESTLVL(R7)      ;INCREMENT PAREN NESTING LEVEL
34 68 B1 016A 364      CMPW  STK_W_TYPE(R8),#OPI_K_OPERAND ;TOP OF STACK OPERAND?
    SF 19 016D 365      BLSS  170$     ;IF GTR NO
    00BB 31 016F 366      BRW   250$     ;REPORT EXPRESSION SYNTAX ERROR
    0172 367 ;
    0172 368 ;
    0172 369 ; CHECK FOR LEGITIMATE UNARY OPERATORS. CONVERT PLUS AND MINUS TO INTEGER
    0172 370 ; NOOP AND NEGATE. PLACE OPERATOR ON STACK.
    0172 371 ;
    0172 372 ;
34 68 B1 0172 373 53$:  CMPW  STK_W_TYPE(R8),#OPI_K_OPERAND ;TOS AN OPERATOR?
    22 18 0175 374      BGEQ  140$     ;IF NO
53 06 B1 0177 375      CMPW  #OPI_K_ADD,R3      ;THIS A UNARY PLUS?
    08 12 017A 376      BNEQ  57$      ;IF NEQ NO, SKIP TO NEXT
53 10 B0 017C 377      MOVW  #OPI_K_POS,R3      ;CONVERT THIS TO INTEGER NOOP
51 0A B0 017F 378      MOVW  #OPP_K_POS,R1      ;
    4A 11 0182 379      BRB   170$     ;GO STACK THIS
53 08 B1 0184 380 57$:  CMPW  #OPI_K_SUB,R3      ;THIS A UNARY MINUS?
    08 12 0187 381      BNEQ  63$     ;IF NEQ NO
53 0E B0 0189 382      MOVW  #OPI_K_NEG,R3      ;CONVERT THIS TO NEGATE
51 0A B0 018C 383      MOVW  #OPP_K_NEG,R1      ;
    3D 11 018F 384      BRB   170$     ;GO STACK THIS
53 02 B1 0191 385 63$:  CMPW  #OPI_K_NOT,R3      ;NOT LPAREN, ADD OR SUB. ONLY LEGAL
    38 13 0194 386      BEQL  170$     ;IF .NOT. STACK OPERATOR
    0094 31 0196 387      BRW   250$     ;GO REPORT SYNTAX ERROR
    0199 388 ;
    0199 389 ;
    0199 390 ; COMPARE OPERATOR PRECEDENCE VALUE WITH OPERATOR IN NEXT TO LAST ITEM ON STACK
    0199 391 ;
    0199 392 ;
34 08 A8 B1 0199 393 140$:  CMPW  STK_W_TYPE+STK_K_LENGTH(R8),#OPI_K_OPERAND ;PREVIOUS ITEM OPERATOR?
    06 18 019D 394      BGEQ  150$     ;IF LEQ NO
51 0A A8 B1 019F 395      CMPW  STK_W_PREC+STK_K_LENGTH(R8),R1 ;STACK OPERATOR HIGHER PRECEDENCE?
    3A 18 01A3 396      BGEQ  190$     ;IF GEQ YES
    53 24 D1 01A5 397 150$:  CMPL  #OPI_K_RPAREN,R3      ;CURRENT OPERATOR RIGHT PARENTHESIS?
    1F 12 01A8 398      BNEQ  160$     ;IF NEQ NO
    67 97 01AA 399      DECB  NESTLVL(R7)      ;DECREMENT PAREN NESTING LEVEL
08 A8 02 B1 01AC 400      CMPW  #OPI_K_NOT,STK_W_TYPE+STK_K_LENGTH(R8) ;OPERATOR BOOLEAN NOT?
    2D 13 01B0 401      BEQL  190$     ;IF EQL YES
08 A8 0E B1 01B2 402      CMPW  #OPI_K_NEG,STK_W_TYPE+STK_K_LENGTH(R8) ;OPERATOR NEGATE?
    27 13 01B6 403      BEQL  190$     ;IF EQL YES
08 A8 10 B1 01B8 404      CMPW  #OPI_K_POS,STK_W_TYPE+STK_K_LENGTH(R8) ;OPERATOR INTEGER NOOP?
    21 13 01BC 405      BEQL  190$     ;IF EQL YES
08 A8 22 B1 01BE 406      CMPW  #OPI_K_LPAREN,STK_W_TYPE+STK_K_LENGTH(R8) ;OPERATOR LEFT PAREN?
    69 12 01C2 407      BNEQ  250$     ;IF NEQ NO
    68 88 7D 01C4 408      MOVQ  (R8)+,(R8)      ;REMOVE LEFT PARENTHESIS FROM STACK
    13 11 01C7 409      BRB   180$     ;
53 20 D1 01C9 410 160$:  CMPL  #OPI_K_EOS,R3      ;END OF STATEMENT OPERATOR?
    69 13 01CC 411      BEQL  270$     ;IF EQL YES
    01CE 412 ;
    01CE 413 ;
    01CE 414 ; STACK ITEM ONTO INTERMEDIATE PARSE STACK
    01CE 415 ;

```

```

78 52 D0 01CE 416 170$: MOVL R2,-(R8) ;STACK NEW ITEM
78 51 B0 01D1 417 :
78 53 B0 01D4 418 : MOVW R1,-(R8)
59 58 D1 01D7 419 : MOVW R3,-(R8)
45 1B 01DA 420 : CMPL R8,R9 ;PARSE STACK OVERFLOW?
FF12 31 01DC 421 180$: BLEQU 240$ ;IF LEQU YES
01DF 422 :
01DF 423 :
01DF 424 :
01DF 425 : UNSTACK TRIAD FROM PARSE STACK
01DF 426 :
01DF 427 :
34 0A BB 01DF 428 190$: PUSHR #*M<R1,R3> ;SAVE OPERATOR PARAMETERS
68 B1 01E1 429 : CMPW STK_W_TYPE(R8),#OPI_K_OPERAND ;TOP ITEM ON STACK OPERAND?
44 19 01E4 430 : BLSS 255$ ;IF GTR NO
54 88 7D 01E6 431 : MOVQ (R8)+,R4 ;REMOVE RIGHT HAND OPERAND FROM STACK
52 88 7D 01E9 432 : MOVQ (R8)+,R2 ;REMOVE OPERATOR FROM STACK
52 02 B1 01EC 433 : CMPW #OPI_K_NOT,R2 ;BOOLEAN NOT?
14 13 01EF 434 : BEQL 220$ ;IF EQL YES
34 68 B1 01F1 435 : CMPW STK_W_TYPE(R8),#OPI_K_OPERAND ;TOP ITEM ON STACK OPERAND?
0C 18 01F4 436 : BGEQ 210$ ;IF YES
52 0E B1 01F6 437 : CMPW #OPI_K_NEG,R2 ;OPERATOR UNARY MINUS?
0A 13 01F9 438 : BEQL 220$ ;IF EQL YES
52 10 B1 01FB 439 : CMPW #OPI_K_POS,R2 ;OPERATOR INTEGER NOOP?
2A 12 01FE 440 : BNEQ 255$ ;IF NEQ NO
03 11 0200 441 : BRB 220$
50 88 7D 0202 442 210$: MOVQ (R8)+,R0 ;REMOVE LEFT HAND OPERAND FROM STACK
78 59 D0 0205 443 220$: MOVL R9,-(R8) ;PUSH ADDRESS OF STACK OPERAND
78 36 9A 0208 444 : MOVZBL #OPI_K_STACK,-(R8) ;PUSH OPERAND INDEX VALUE
58 59 D1 020B 445 : CMPL R9,R8 ;TRIAD STACK OVERFLOW?
0E 1A 020E 446 : BGTRU 245$ ;IF GTRU YES
89 54 7D 0210 447 : MOVQ R4,(R9)+ ;PUSH RIGHT HAND OPERAND
89 52 3C 0213 448 : MOVZWL R2,(R9)+ ;PUSH OPERATOR INDEX
89 50 7D 0216 449 : MOVQ R0,(R9)+ ;PUSH LEFT HAND OPERAND
0A BA 0219 450 230$: POPR #*M<R1,R3> ;RETRIEVE OPERATOR PARAMETERS
FF7B 31 021B 451 : BRW 140$
021E 452 :
021E 453 :
021E 454 : EXPRESION TOO COMPLEX
021E 455 :
021E 456 :
5E 08 C0 021E 457 245$: ADDL #8,SP ;POP SAVED R1,R3 OFF STACK
0A 11 0221 458 240$: STATUS COMPLX ;SET COMPLEX EXPRESION STATUS
0228 459 :
022A 460 :
022A 461 :
022A 462 : EXPRESION SYNTAX ERROR
022A 463 :
022A 464 :
5E 08 C0 022A 465 255$: ADDL #8,SP ;POP SAVED R1,R3 OFF STACK
022D 466 250$: STATUS EXPSYN ;SET EXPRESION SYNTAX ERROR STATUS
01AF 31 0234 467 260$: BRW EXIT ;AND EXIT
0237 468 :
0237 469 :
0237 470 : END OF STATEMENT - CHECK FOR VALID PARSE
0237 471 :
0237 472 :

```

- EXPRESSION ANALYSIS  
EXPRESSION ANALYSIS

34	68	B1	0237	473	270\$:	CMPW	STK_W_TYPE(R8),#OPI_K_OPERAND ;TOP ITEM ON STACK OPERAND?
	F1	19	023A	474		BLSS	250\$ ;ERROR IF NOT
89	88	7D	023C	475		MOVQ	(R8)+,(R9)+ ;MOVE RIGHT HAND OPERAND TO TRIAD STACK
68	1E	B1	023F	476		CMPW	#OPI_K_SOS,STK_W_TYPE(R8) ;TOP OF STACK START OF STATEMENT?
	E9	12	0242	477		BNEQ	250\$ ;IF NEQ NO
69	26	9A	0244	478		MOVZBL	#OPI_K_STORE,(R9) ;INSERT STORE OPERATOR
	00B9	31	0247	479		BRW	EVALDATE ;EVALUATE THE EXPRESSION

```

024A 481 .SBTTL SEARCH OPERATOR TABLE
024A 482 :+
024A 483 : FIND_OPERATOR - SEARCH OPERATOR TABLE
024A 484 :
024A 485 : THIS ROUTINE SEARCHES THE OPERATOR TABLE, GIVEN A STRING, TO
024A 486 : LOCATE THE OPERATOR PRECEDENCE AND INDEX.
024A 487 :
024A 488 : INPUTS:
024A 489 :
024A 490 : R1/R2 = DESCRIPTOR OF STRING
024A 491 :
024A 492 : OUTPUTS:
024A 493 :
024A 494 : R0 = STATUS
024A 495 : R1 = OPERATOR PRECEDENCE
024A 496 : R3 = OPERATOR INDEX
024A 497 :
024A 498 : R4,R5 DESTROYED.
024A 499 :
024A 500 FIND_OPERATOR:
50 FDB2 CF 9E 024A 501 MOVAB OPTAB,R0 ;GET ADDRESS OF OPERATOR TABLE
55 50 D0 024F 502 40$: MOVL R0,R5 ;RETRIEVE ADDRESS OF NEXT OPERATOR ENTRY
50 85 9A 0252 503 MOVZBL (R5)+,R0 ;GET OFFSET TO NEXT OPERATOR ENTRY
50 1D 13 0255 504 BEQL 80$ ;IF EQL END OF TABLE
50 55 C0 0257 505 ADDL R5,R0 ;CALCULATE ADDRESS OF NEXT ENTRY
85 51 91 025A 506 CMPB R1,(R5)+ ;LENGTH OF NAMES MATCH?
50 F0 12 025D 507 BNEQ 40$ ;IF NEQ NO
53 51 7D 025F 508 MOVQ R1,R3 ;COPY OPERATOR DESCRIPTOR
85 84 91 0262 509 50$: CMPB (R4)+,(R5)+ ;CHARACTERS MATCH?
85 E8 12 0265 510 BNEQ 40$ ;IF NEQ NO
F8 53 F5 0267 511 SOBGTR R3,50$ ;ANY MORE CHARACTERS TO MATCH?
51 85 9A 026A 512 MOVZBL (R5)+,R1 ;GET OPERATOR PRECEDENCE VALUE
53 65 9A 026D 513 MOVZBL (R5),R3 ;GET OPERATOR INDEX VALUE
50 01 D0 0270 514 MOVL #1,R0 ;SUCCESS
05 0273 515 RSB
0274 516 80$: STATUS IVOPER ;REPORT INVALID OPERATOR
05 027B 517 RSB

```



```

027C 519          .SBTTL  PARSE_OPERAND - PARSE OPERAND TOKEN
027C 520          :+
027C 521          : PARSE_OPERAND - PARSE OPERAND TOKEN
027C 522          :
027C 523          : THIS ROUTINE IS CALLED TO PARSE AN OPERAND TOKEN AND RETURN
027C 524          : THE OPERAND PARAMETERS DESCRIBING THE OPERAND.
027C 525          :
027C 526          : INPUTS:
027C 527          :
027C 528          : R1/R2 = STRING DESCRIPTOR OF OPERAND
027C 529          :
027C 530          : OUTPUTS:
027C 531          :
027C 532          : R0 = STATUS
027C 533          :
027C 534          : THE CALLER'S STACK MAY BE UPDATED IN ORDER TO SAVE
027C 535          : INTERMEDIATE STRINGS OR BINARY VALUES.
027C 536          :-
027C 537
027C 538 PARSE_OPERAND:
027C 539 MOVZBL (R2),-(SP) ;SAVE STARTING CHARACTER OF SYMBOL
027F 540 BSBW DCL$COMPRESS ;COMPRESS QUOTED STRING
53 FD7E' 30 0282 541 MOVZBL #OPI_K_STRING,R3 ;ASSUME CHARACTER STRING LITERAL
8E 22 D1 0285 542 CMPL #^A/^/,(SP)+ ;STRING LITERAL?
54 13 0288 543 BEQL 170$ ;IF EQL YES
62 25 91 028A 544 CMPB #^A/%/, (R2) ;NUMERIC RADIX OPERATOR?
53 13 028D 545 BEQL 105$ ;BR IF YES-ITS A NUMBER
62 30 91 028F 546 CMPB #^A/0/, (R2) ;POSSIBLY NUMERIC CONSTANT?
05 1A 0292 547 BGTRU 90$ ;IF GTRU NO
62 39 91 0294 548 CMPB #^A/9/, (R2) ;NUMERIC CONSTANT?
49 1E 0297 549 BGEQU 105$ ;IF GEQU YES
57 DD 0299 550 90$: PUSHL R7 ;SAVE ADDRESS OF STACK STORAGE
0C E1 029B 551 BBC #WRK V COMMENT, - ;SKIP IF NOT IN COMMENT
03 FO AA 029D 552 WRK Q FLAGS(R10),92$ ;MAKE FSVER EXPR EVAL WORK IN COMMENTS
FD5D' 30 02A0 553 BSBW DCL$UPCASE ;UPCASE THE SYMBOL NAME
56 51 7D 02A3 554 92$: MOVQ R1,R6 ;COPY DESCRIPTOR FOR DCL$LEXIF
FD57' 30 02A6 555 BSBW DCL$SEARCH ;SEARCH SYMBOL TABLE FOR VALUE
03 50 E8 02A9 556 BLBS R0,95$ ;IF LBC SEARCH FAILURE
FD51' 30 02AC 557 BSBW DCL$LEXIF ;GET VALUE OF LEXICAL FUNCTION
57 8ED0 02AF 558 95$: POPL R7 ;RESTORE ADDRESS OF STACK STORAGE
46 50 E9 02B2 559 BLBC R0,110$ ;BRANCH IF NOT A FUNCTION
52 D5 02B5 560 TSTL R2 ;BINARY OR STRING?
17 13 02B7 561 BEQL 96$ ;BRANCH IF BINARY
50 8ED0 02B9 562 POPL R0 ;PICK UP RETURN ADDRESS
5E 51 C2 02BC 563 SUBL R1,SP ;ALLOCATE SPACE FOR STRING ON STACK
50 DD 02BF 564 PUSHL R0 ;RESTORE RETURN ADDRESS
56 51 D0 02C1 565 MOVL R1,R6 ;SAVE STRING SIZE
04 AE 62 51 28 02C4 566 MOVC R1,(R2),4(SP) ;COPY STRING OPERAND TO STACK
51 56 D0 02C9 567 MOVL R6,R1 ;RESTORE STRING SIZE
52 04 AE 9E 02CC 568 MOVAB 4(SP),R2 ;SET ADDRESS OF COPIED STRING
50 8ED0 02D0 569 96$: POPL R0 ;PICK UP RETURN ADDRESS
7E 51 7D 02D3 570 MOVQ R1,-(SP) ;PUSH VALUE DESCRIPTOR ONTO STACK
52 5E D0 02D6 571 MOVL SP,R2 ;SET ADDRESS OF ON-STACK OPERAND
50 DD 02D9 572 PUSHL R0 ;RESTORE RETURN ADDRESS
53 36 9A 02DB 573 MOVZBL #OPI_K_STACK,R3 ;SET TYPE OF ITEM
50 01 D0 02DE 574 170$: MOVL #1,R0 ;SUCCESS
05 05 02E1 575 RSB

```

			02E2	576	:						
			02E2	577	:	PARSE	NUMERIC	OPERAND			
			02E2	578	:						
51	52	51	7D	02E2	579	105\$:	MOVQ	R1,R2			:COPY DESCRIPTOR
	00AE	CB	9A	02E5	580		MOVZBL	PRC B DEFRADIX(R11),R1			:SET RADIX FOR CONVERSION
		FD13	30	02EA	581		BSBW	DCL\$CNVASCBIN			:CONVERT TO BINARY LONGWORD
		04	12	02ED	582		BNEQ	108\$			:BRANCH IF CONVERSION ERROR
		52	D4	02EF	583		CLRL	R2			:MARK VALUE IS BINARY
		DD	11	02F1	584		BRB	96\$			:AND STORE IT ON THE STACK
				02F3	585						
				02F3	586	108\$:	STATUS	IVCHAR			:ERROR CONVERTING NUMBER
			05	02FA	587		RSB				
				02FB	588	110\$:	STATUS	UNDSYM			:SET UNDEFINED SYMBOL STATUS
			05	0302	589		RSB				

```

0303 591 :
0303 592 : EVALUATE THE EXPRESSION WHICH NOW CONSISTS OF TRIADS ORDERED
0303 593 : BY PRECEDENCE, DESCRIBING THE ARGUMENTS AND OPERATORS IN THE
0303 594 : ORDER THEY ARE TO BE EVALUATED.
0303 595 :
0303 596 :
0303 597 EVALUATE:
59 FE10 C7 9E 0303 598 :EVALUATE EXPRESSION
5C 59 DO 0308 599 10$: MOVAB -TRIADSTKSIZ-PARSESTKSIZ(R7),R9 ;GET STARTING ADDRESS OF TRIAD STACK
00E9 30 0308 600 10$: MOVL R9,AP ;ADDRESS TO STORE RESULT
56 89 DO 030E 601 BSBW FETCH ;FETCH RIGHT HAND OPERAND PARAMETERS
28 56 D1 0311 602 MOVL (R9)+,R6 ;FETCH OPERATOR INDEX
5A 18 0314 603 CMPL R6,#OPI_K_EQS ;STRING OPERATOR?
26 56 D1 0316 604 BGEQ 20$ ;IF GEQ YES
53 13 0319 605 CMPL R6,#OPI_K_STORE ;STORE OPERATOR?
06 56 D1 031B 606 BEQL 40$ ;IF EQL YES
08 05 13 031E 607 CMPL R6,#OPI_K_ADD ;ADD?
26 12 D1 0320 608 BEQL 12$ ;BRANCH IF SO
0323 609 CMPL R6,#OPI_K_SUB ;SUB?
0325 610 BNEQ 15$ ;BRANCH IF NOT
0325 611 :
0325 612 : (+) OR (-) OPERATORS. DECIDE IF STRING OR ARITHMETIC OPERATOR SHOULD BE
0325 613 : APPLIED. ASSUME ARITHMETIC UNLESS BOTH SIDES ARE STRINGS
0325 614 :
0325 615 :
00D9 30 0325 616 12$: BSBW OPERAND ;GET RIGHT HAND OPERAND DESCRIPTOR
21 13 0328 617 BEQL 15$ ;IF BINARY, DO AN INTEGER ADD
54 51 7D 032A 618 MOVQ R1,R4 ;SAVE RIGHT HAND OPERAND STRING DESCRIPTOR
00C7 30 032D 619 BSBW FETCH ;FETCH LEFT HAND OPERAND PARAMETERS
00CE 30 0330 620 BSBW OPERAND ;GET LEFT HAND OPERAND DESCRIPTOR
OD 13 0333 621 BEQL 14$ ;IF BINARY, ATTEMPT ARITHMETIC OPERATION
52 51 7D 0335 622 MOVQ R1,R2 ;SAVE LEFT HAND OPERAND DESCRIPTOR
044D'CF46 16 0338 623 JSB W^STRINGDISP-OPI_K_ADD[R6] ;PERFORM STRING OPERATION
033D 624 ;(RESULTANT STRING PLACED ONTO STACK)
6C 54 7D 033D 625 MOVQ R4,(AP) ;STORE RESULTANT STRING DESCRIPTOR
C6 11 0340 626 BRB 10$
59 14 C2 0342 627 14$: SUBL #8+4+8,R9 ;BACKUP TRIAD STACK
00AF 30 0345 628 BSBW FETCH ;RE-FETCH LEFT HAND OPERAND
56 89 DO 0348 629 MOVL (R9)+,R6 ;RE-FETCH OPERATOR INDEX
034B 630 :
034B 631 :
034B 632 : ARITHMETIC RELATIONAL OR BOOLEAN OPERATOR
034B 633 :
034B 634 :
00CE 30 034B 635 15$: BSBW NUMERIC ;CONVERT RIGHT HAND OPERAND TO NUMERIC
54 50 DO 034E 636 MOVL R0,R4 ;SAVE RIGHT HAND OPERAND VALUE
00A3 30 0351 637 BSBW FETCH ;FETCH LEFT HAND OPERAND PARAMETERS
02 56 D1 0354 638 CMPL R6,#OPI_K_NOT ;BOOLEAN NOT OPERATOR?
44 13 0357 639 BEQL 30$ ;IF EQL YES
0E 56 D1 0359 640 CMPL R6,#OPI_K_NEG ;INTEGER NEGATION?
3F 13 035C 641 BEQL 30$ ;IF EQL YES
10 56 D1 035E 642 CMPL R6,#OPI_K_POS ;INTEGER NOOP?
3A 13 0361 643 BEQL 30$ ;IF EQL YES
00B6 30 0363 644 BSBW NUMERIC ;CONVERT LEFT HAND OPERAND TO NUMERIC
52 50 DO 0366 645 MOVL R0,R2 ;SAVE LEFT HAND OPERAND VALUE
54 52 D1 0369 646 CMPL R2,R4 ;COMPARE RIGHT AND LEFT HAND OPERANDS
2F 11 036C 647 BRB 30$

```

```

3B 11 036E 648
      036E 649 40$: BRB 41$: ;CONTINUE TO STORE RESULT CODE
      0370 650 ;
      0370 651 ; STRING RELATIONAL OPERATOR
      0370 652 ;
      0370 653 ;
      009C 30 0370 654 20$: BSBW STRING ;CONVERT RIGHT HAND OPERAND TO STRING
      F486 CA DD 0373 655 PUSHL WRK_L_EXPANDPTR(R10) ;ASSUME RESULT IS IN EXPANSION BUFFER
      F486 CA 52 D1 0377 656 CMPL R2,WRK_L_EXPANDPTR(R10) ;IS RESULT IN EXPANSION BUFFER
      05 12 037C 657 BNEQ 25$ ;IF NOT, THEN SKIP
      F486 CA 51 C0 037E 658 ADDL R1,WRK_L_EXPANDPTR(R10) ;MOVE EXPANSION POINTER PAST STRING
      54 51 7D 0383 659 25$: MOVQ R1,R4 ;SAVE RIGHT HAND OPERAND STRING DESCRIPTOR
      006E 30 0386 660 BSBW FE1CH ;FETCH LEFT HAND OPERAND PARAMETERS
      0083 30 0389 661 BSBW STRING ;CONVERT LEFT HAND OPERAND TO STRING
      52 51 7D 038C 662 MOVQ R1,R2 ;SAVE LEFT HAND OPERAND DESCRIPTOR
      56 16 C2 038F 663 SUBL #OPI,K EQS-OPI,K EQ,R6 ;NORMALIZE OPERATOR INDEX
      F486 CA 8ED0 0392 664 POPL WRK_L_EXPANDPTR(R10) ;RESTORE EXPANSION BUFFER POINTER
      65 54 00 63 52 2D 0397 665 CMPCS R2,(R3),#0,R4,(R5) ;COMPARE RIGHT AND LEFT HAND OPERANDS
      04C2'CF46 16 039D 666 30$: JSB W^DISPATCH[R6] ;EXECUTE OPERATOR SPECIFIC ROUTINE
      6C 54 D0 03A2 667 MOVL R4,(AP) ;STORE OPERATION RESULT
      04 AC D4 03A5 668 CLRL 4(AP) ;INDICATE RESULT IS BINARY
      FF5D 31 03A8 669 BRW 10$ ;
      03AB 670 ;
      03AB 671 ;
      03AB 672 ; STORE RESULT OPERATOR
      03AB 673 ;
      54 10 03AB 674 41$: BSBW OPERAND ;GET DESCRIPTOR OF RIGHT-HAND OPERAND
      30 13 03AD 675 BEQL 50$ ;BRANCH IF BINARY RESULT
      50 50 F492 CA 9E 03AF 676 MOVAB WRK_G_BUFFER(R10),R0 ;GET ADDRESS OF EXPANSION BUFFER
      00000400 8F C0 03B4 677 ADDL #WRK_C_CMDBUFSIZ,R0 ;FIND END OF BUFFER
      50 F486 CA C2 03BB 678 SUBL WRK_L_EXPANDPTR(R10),R0 ;CALCULATE HOW MUCH IS LEFT
      51 50 D1 03C0 679 CMPL R0,R1 ;WILL RESULT FIT?
      09 14 03C3 680 BGTR 45$ ;YES, THEN STORE IT AWAY
      03C5 681 STATUS BUFOVF ;SET OVERFLOW STATUS
      18 11 03CC 682 BRB EXIT ;EXIT WITH ERROR
      F486 DA 56 51 D0 03CE 683 45$: MOVL R1,R6 ;SAVE LENGTH OF STRING
      62 51 28 03D1 684 MOVC R1,(R2),@WRK_L_EXPANDPTR(R10) ;COPY IT TO THE EXPANSION BUFFER
      51 56 D0 03D7 685 MOVL R6,R1 ;RETURN LENGTH OF RESULT STRING
      52 F486 CA D0 03DA 686 MOVL WRK_L_EXPANDPTR(R10),R2 ;AND ADDRESS OF RESULT STRING
      03DF 687 50$: STATUS NORMAL ;SET NORMAL COMPLETION STATUS
      SE 0C A7 9E 03E6 688 EXIT: MOVAB LOCALSIZ(R7),SP ;DEALLOCATE SCRATCH STACKS
      13F8 8F BA 03EA 689 POPR #*M<R3,R4,R5,R6,R7,R8,R9,AP> ;RESTORE REGISTERS
      05 50 E9 03EE 690 BLBC R0,90$ ;BRANCH IF ERROR EXIT
      F48A CA 53 D0 03F1 691 MOVL R3,WRK_L_MARKPTR(R10) ;LEAVE MARKPTR SET TO START OF EXPR.
      05 03F6 692 90$: RSB ;

```

```
03F7 694 .SBTTL FETCH NEXT OPERAND FROM TRIAD STACK
03F7 695 :
03F7 696 : FETCH - FETCH NEXT OPERAND FROM TRIAD STACK
03F7 697 :
03F7 698 :
50 89 3C 03F7 699 FETCH: MOVZWL (R9)+,R0 :GET TYPE OF OPERAND
51 89 3C 03FA 700 MOVZWL (R9)+,R1 :GET LENGTH OF OPERAND
52 89 D0 03FD 701 MOVL (R9)+,R2 :GET ADDRESS OF OPERAND
05 0400 702 RSB :
```

```
0401 704 .SBTTL GET OPERAND DESCRIPTOR
0401 705 :
0401 706 : OPERAND - GET OPERAND DESCRIPTOR FROM OPERAND PARAMETERS
0401 707 :
0401 708 : Z BIT SET ON R2 (ADDRESS), Z=1 INDICATES BINARY, Z=0 INDICATES STRING
0401 709 :
0401 710 OPERAND:
50 36 D1 0401 711 CML #OPI_K_STACK,R0 ;STACK OPERAND?
06 12 0404 712 BNEQ 10$ ;BRANCH IF NOT
51 62 7D 0406 713 MOVQ (R2),R1 ;GET OPERAND DESCRIPTOR
50 34 9A 0409 714 MOVZBL #OPI_K_STRING,R0 ;MARK OPERAND NOW A STRING
52 05 D5 040C 715 10$: TSTL R2 ;SET CONDITION CODES ON TYPE
05 040E 716 RSB
```

```
040F 718 .SBTTL CONVERT STRING PARAMETERS TO STRING OPERAND
040F 719 :
040F 720 : STRING - CONVERT OPERAND TO CHARACTER STRING
040F 721 :
040F 722 : IF THE OPERAND IS EVALUATED, THEN RETURN ITS DESCRIPTOR IF ITS
040F 723 : A STRING, AND IF NUMERIC, CONVERT IT TO ASCII DECIMAL.
040F 724 : IF THE OPERAND IS AN UNEVALUATED CHARACTER OR NUMERIC STRING LITERAL,
040F 725 : SIMPLY RETURN WITH THE DESCRIPTOR.
040F 726 :
040F 727 :
040F 728 STRING: ;STRING OPERAND
FO 10 040F 729 BSBB OPERAND ;GET OPERAND DESCRIPTOR
08 12 0411 730 BNEQ 10$ ;IF STRING, EXIT WITH DESCRIPTOR
53 DD 0413 731 PUSHL R3 ;SAVE R3
FBE8' 30 0415 732 BSBW DCL$CVT_STRING ;CONVERT TO STRING
53 8ED0 0418 733 POPL R3 ;RESTORE R3
05 041B 734 10$: RSB ;
```

```

041C 736      .SBTTL  CONVERT OPERAND PARAMETERS TO NUMERIC VALUE
041C 737      :
041C 738      : NUMERIC - CONVERT OPERAND PARAMETERS TO NUMERIC VALUE
041C 739      :
041C 740      :
041C 741      NUMERIC:                ;NUMERIC OPERAND
041C 742      :
041C 743      : IF THE OPERAND IS ALREADY BINARY FROM A PREVIOUS EVALUATION, RETURN IT
041C 744      :
      E3 10 041C 745      BSBB  OPERAND          ;GET OPERAND DESCRIPTOR
      1A 13 041E 746      BEQL  30$            ;IF BINARY, RETURN WITH VALUE IN R1
0420 747      :
0420 748      : CONVERT A NUMERIC STRING TO A BINARY LONGWORD
0420 749      :
7E 00AE CB 9A 0420 750      MOVZBL PRC_B_DEFRADIX(R11),-(SP) ;SAVE DEFAULT RADIX
      50 34 D1 0425 751      CMPL  #OPT_R_STRING,R0          ;IS THIS A CHARACTER STRING LITERAL?
      05 13 0428 752      BEQL  10$            ;IF SO, USE DEFAULT RADIX
00AE CB 01 9A 042A 753      MOVZBL #PRC_K_DEC,PRC_B_DEFRADIX(R11) ;OTHERWISE, USE DECIMAL RADIX
      FC65 30 042F 754 10$:  BSBW  DCL$CVT_BINARY          ;CONVERT STRING TO BINARY
00AE CB 8E F6 0432 755      CRTL  B (SP)+,PRC_B_DEFRADIX(R11) ;RESTORE DEFAULT RADIX
      04 50 E9 0437 756      BLBC  R0,85$          ;BRANCH IF ERROR DETECTED
      50 51 D0 043A 757 30$:  MOVL  R1,R0            ;SET VALUE
      05 043D 758      RSB
      043E 759      :
      FBBF' 30 043E 760 85$:  BSBW  DCL$MARK          ;MARK START OF ERROR SEGMENT
      50 62 90 0441 761      MOVB  (R2),R0          ;PICK UP FIRST BYTE OF STRING
      FBB9' 30 0444 762      BSBW  DCL$PUTCHAR        ;WRITE CHARACTER INTO BUFFER
      0447 763      :
      0447 764      : SO THAT ERROR REPORTING CAN
      8E D5 0447 765      TSTL  (SP)+          ;DISPLAY THE CHARACTER
      0449 766      STATUS IVCHAR          ;CLEAN STACK OF PREVIOUS CALLER
      FF93 31 0450 767      BRW  EXIT            ;INVALID CHARACTER
      :
      : ;RETURN TO CALLER'S CALLER

```



```
0453 769 .SBTTL DISPATCH STRING OPERATION FUNCTION
0453 770 :
0453 771 : DISPATCH FUNCTIONS WHICH RESULT IN STRINGS
0453 772 :
0453 773 STRINGDISP:
02 11 0453 774 BRB CONCAT ;STRING CONCATENATION
36 11 0455 775 BRB REDUCE ;STRING REDUCTION
```

```

0457 777      .SBTTL  STRING CONCATENATION OPERATOR
0457 778      :
0457 779      : THIS ROUTINE PROCESSES THE STRING CONCATENATION OPERATOR AND STORES
0457 780      : THE RESULTANT STRING ON THE CALLER'S STACK.
0457 781      :
0457 782      : INPUTS:
0457 783      :
0457 784      :     R2/R3 = STRING DESCRIPTOR OF LEFT-HAND SIDE
0457 785      :     R4/R5 = STRING DESCRIPTOR OF RIGHT-HAND SIDE
0457 786      :
0457 787      : OUTPUTS:
0457 788      :
0457 789      :     R4/R5 = STRING DESCRIPTOR OF RESULTANT STRING
0457 790      :
0457 791      :     R0-R3,R6 DESTROYED.
0457 792      :
0457 793      : CONCAT:  POPL   R6           ;POP CALLER'S RETURN ADDRESS
0457 794      :          ADDL3  R2,R4,R0      ;COMPUTE SIZE OF RESULTANT STRING
0457 795      :          SUBL   R0,SP          ;ALLOCATE SPACE ON CALLER'S STACK
0457 796      :          PROBEW #0,#128,-128(SP) ;CHECK FOR STACK OVERFLOW
0457 797      :          BNEQ   10$           ;CONTINUE IF SUCCESSFUL
0457 798      :          STATUS BUFOVF        ;SET STATUS
0457 799      :          BRW    EXIT          ;EXIT WITH STATUS
0457 800      :          PUSHL  R0           ;SAVE SIZE OF RESULTANT STRING
0457 801      :          MOVQ   R4,-(SP)       ;SAVE DESCRIPTOR OF RIGHT-SIDE
0457 802      :          MOVC   R2,(R3),12(SP) ;COPY FIRST STRING ONTO STACK
0457 803      :          MOVQ   (SP)+,R4       ;RESTORE DESCRIPTOR OF RIGHT-SIDE
0457 804      :          MOVC   R4,(R5),(R3)   ;APPEND SECOND STRING TO FIRST
0457 805      :          POPL   R4           ;RESTORE SIZE OF RESULTANT STRING
0457 806      :          MOVL  SP,R5          ;SET ADDRESS OF RESULTANT STRING
0457 807      :          JMP   (R6)          ;RETURN

```

```

      56 BED0
      52 C1
      50 C2
80 AE 0080 8F 00 0D
      0A 12
      FF72 31
      50 DD
      7E 54 7D
OC AE 63 52 28
      54 8E 7D
      63 65 54 28
      54 BED0
      55 5E D0
      66 17 0488

```

```

048D 809      .SBTTL STRING REDUCTION OPERATOR
048D 810      :
048D 811      : THIS ROUTINE PROCESSES THE STRING REDUCTION OPERATOR (-) AND STORES
048D 812      : THE RESULTANT STRING ON THE CALLER'S STACK.
048D 813      :
048D 814      : INPUTS:
048D 815      :
048D 816      : R2/R3 = STRING DESCRIPTOR OF LEFT-HAND SIDE
048D 817      : R4/R5 = STRING DESCRIPTOR OF RIGHT-HAND SIDE
048D 818      :
048D 819      : OUTPUTS:
048D 820      :
048D 821      : R4/R5 = STRING DESCRIPTOR OF RESULTANT STRING
048D 822      :
048D 823      : R0-R3,R6 DESTROYED.
048D 824      :
63  52  65  0C  BB 048D 825 REDUCE: PUSHR #*M<R2,R3>          ;SAVE LEFT SIDE OPERAND
      54  39 048F 826 MATCHC R4,(R5),R2,(R3)          ;LOCATE SUBSTRING WITHIN STRING
      28  12 0494 827 BNEQ 50$                      ;BRANCH IF NOT FOUND IN LEFT SIDE
      50  52 7D 0496 828 MOVQ R2,R0                  ;SAVE DESCRIPTOR OF PIECE AFTER MATCH
      004C 8F BA 0499 829 POPR #*M<R2,R3,R6>          ;GET LEFT SIDE AND RETURN ADDRESS
54  52 54 C3 049D 830 SUBL3 R4,R2,R4                ;COMPUTE SIZE OF RESULTANT STRING
52  54 50 C3 04A1 831 SUBL3 R0,R4,R2                ;COMPUTE SIZE OF PIECE BEFORE MATCH
      5E 54 C2 04A5 832 SUBL R4,SP                  ;ALLOCATE SPACE ON CALLER'S STACK
      13 BB 04A8 833 PUSHR #*M<R0,R1,R4>            ;SAVE REGISTERS
OC AE 63 52 28 04AA 834 MOVQ R2,(R3),12(SP)          ;COPY PIECE BEFORE MATCH INTO RESULT
      50 8E 7D 04AF 835 MOVQ (SP)+,R0                ;GET DESCRIPTOR OF PIECE AFTER MATCH
63  61 50 28 0482 836 MOVQ R0,(R1),(R3)            ;APPEND PIECE AFTER MATCH INTO RESULT
      54 8ED0 0486 837 POPL R4                       ;RESTORE SIZE OF RESULTANT STRING
      55 5E D0 0489 838 MOVL SP,R5                  ;SET ADDRESS OF RESULTANT STRING
      66 17 048C 839 JMP (R6)                        ;RETURN
      04BE 840 :
      04BE 841 : SUBSTRING NOT FOUND IN STRING - RETURN MAIN STRING INTACT
      04BE 842 :
      54 8E 7D 04BE 843 50$: MOVQ (SP)+,R4            ;RETURN DESCRIPTOR OF LEFT SIDE
      05 04C1 844 RSB

```

```

04C2 846 .SBTTL DISPATCH BINARY/LOGICAL OPERATOR
04C2 847 :
04C2 848 : DISPATCH BINARY/LOGICAL OPERATOR FUNCTION
04C2 849 :
04C2 850 : INPUTS:
04C2 851 :
04C2 852 : R2/R3 = LEFT HAND OPERAND
04C2 853 : R4/R5 = RIGHT HAND OPERAND
04C2 854 :
04C2 855 : OUTPUTS:
04C2 856 :
04C2 857 : R4/R5 = RESULT VALUE
04C2 858 :
04C2 859 :
04C2 860 DISPATCH:
1E 11 04C2 861 BRB AND ;BOOLEAN AND
23 11 04C4 862 BRB NOT ;BOOLEAN NOT
25 11 04C6 863 BRB OR ;BOOLEAN OR
27 11 04C8 864 BRB ADD ;INTEGER ADD
29 11 04CA 865 BRB SUB ;INTEGER SUBTRACT
2C 11 04CC 866 BRB MUL ;INTEGER MULTIPLY
2E 11 04CE 867 BRB DIV ;INTEGER DIVIDE
41 11 04D0 868 BRB NEG ;INTEGER NEGATION
43 11 04D2 869 BRB POS ;INTEGER NOOP
42 11 04D4 870 BRB EQL ;EQUAL
44 11 04D6 871 BRB GEQ ;GREATER OR EQUAL
46 11 04D8 872 BRB GTR ;GREATER
48 11 04DA 873 BRB LEQ ;LESS OR EQUAL
4A 11 04DC 874 BRB LSS ;LESS
04DE 875 : BRB NEQ ;NOT EQUAL
04DE 876 :
04DE 877 :
04DE 878 : NOT EQUAL
04DE 879 :
04DE 880 :
4D 12 04DE 881 NEQ: BNEQ SETTRUE ;IF NEQ SETTRUE RESULT
48 11 04E0 882 BRB SETFALSE
04E2 883 :
04E2 884 :
04E2 885 : BOOLEAN AND
04E2 886 :
04E2 887 :
52 52 D2 04E2 888 AND: MCOML R2,R2 ;COMPLEMENT LEFT HAND OPERAND
54 52 CA 04E5 889 BICL R2,R4 ;FORM BOOLEAN AND FUNCTION
05 04E8 890 RSB ;
04E9 891 :
04E9 892 :
04E9 893 : BOOLEAN NOT
04E9 894 :
04E9 895 :
54 54 D2 04E9 896 NOT: MCOML R4,R4 ;FORM BOOLEAN NOT FUNCTION
05 04EC 897 RSB ;
04ED 898 :
04ED 899 :
04ED 900 : BOOLEAN OR
04ED 901 :
04ED 902 :

```

- EXPRESSION ANALYSIS  
DISPATCH BINARY/LOGICAL OPERATOR

M 3

15-SEP-1984 23:46:42 VAX/VMS Macro V04-00  
4-SEP-1984 23:40:31 [DCL.SRC]EXPRESS.MAR;1

54	52	C8	04ED	903	OR:	BISL	R2,R4	:FORM BOOLEAN OR FUNCTION
		05	04F0	904		RSB		:
			04F1	905				:
			04F1	906				:
			04F1	907	:	INTEGER ADD		:
			04F1	908	:			:
			04F1	909	:			:
54	52	C0	04F1	910	ADD:	ADDL	R2,R4	:FORM ARITHMETIC SUM
		05	04F4	911		RSB		:
			04F5	912				:
			04F5	913	:	INTEGER SUBTRACT		:
			04F5	914	:			:
			04F5	915	:			:
			04F5	916	:			:
54	52	54	C3	04F5	917	SUB:	SUBL3 R4,R2,R4	:FORM ARITHMETIC DIFFERENCE
			05	04F9	918		RSB	:
				04FA	919			:
				04FA	920	:	INTEGER MULTIPLY	:
				04FA	921	:		:
				04FA	922	:		:
				04FA	923	:		:
54	52	C4	04FA	924	MUL:	MULL	R2,R4	:FORM ARITHMETIC PRODUCT
		05	04FD	925		RSB		:
			04FE	926				:
			04FE	927	:	INTEGER DIVIDE		:
			04FE	928	:			:
			04FE	929	:			:
			04FE	930	:			:
	54	D5	04FE	931	DIV:	TSTL	R4	:DIVIDE BY ZERO ATTEMPT?
	0C	12	0500	932		BNEQ	20\$	:BR IF NO
			0502	933		SETBIT	#31,R4	:MAKE RESULT THE HIGHEST NEGATIVE NUMBER
	52	D5	0506	934		TSTL	R2	:SOURCE NEGATIVE?
	03	19	0508	935		BLSS	10\$	:BR IF YES
	54	54	D2	050A	936	MCOML	R4,R4	:MAKE THE LARGEST POSITIVE NUMBER
			05	050D	937	10\$:	RSB	:
54	52	54	C7	050E	938	20\$:	DIVL3 R4,R2,R4	:FORM ARITHMETIC QUOTIENT
			05	0512	939		RSB	:
				0513	940			:
				0513	941			:
				0513	942	:	INTEGER NEGATION	:
				0513	943	:		:
				0513	944	:		:
				0513	945	:		:
54	54	CE	0513	946	NEG:	MNEGL	R4,R4	:NEGATE OPERAND
		05	0516	947		RSB		:
			0517	948				:
			0517	949	:	INTEGER NOOP		:
			0517	950	:			:
			0517	951	:			:
			0517	952	:			:
		05	0517	953	POS:	RSB		:NOOP
			0518	954				:
			0518	955	:	EQUAL		:
			0518	956	:			:
			0518	957	:			:
			0518	958	:			:
	13	13	0518	959	EQL:	BEQL	SETRUE	:IF EQL SETTRUE RESULT

```

OE 11 051A 960 BRB SETFALSE
      051C 961
      051C 962 :
      051C 963 : GREATER OR EQUAL
      051C 964 :
      051C 965
OF 18 051C 966 GEQ: BGEO SETTRUE ;IF GEQ SETTRUE RESULT
OA 11 051E 967 BRB SETFALSE
      0520 968
      0520 969 :
      0520 970 : GREATER
      0520 971 :
      0520 972
OB 14 0520 973 GTR: BGTR SETTRUE ;IF GTR SETTRUE RESULT
OA 11 0522 974 BRB SETFALSE
      0524 975
      0524 976 :
      0524 977 : LESS OF EQUAL
      0524 978 :
      0524 979
O7 15 0524 980 LEQ: BLEQ SETTRUE ;IF LEQ SETTRUE RESULT
O2 11 0526 981 BRB SETFALSE
      0528 982
      0528 983 :
      0528 984 : LESS
      0528 985 :
      0528 986
O3 19 0528 987 LSS: BLSS SETTRUE ;IF LSS SETTRUE RESULT
      052A 988
      052A 989 :
      052A 990 : RETURN FALSE RESULT
      052A 991 :
      052A 992
      052A 993 SETFALSE:
54 D4 052A 994 CLRL R4 ;SET RESULT FALSE
O5 05 052C 995 RSB
      052D 996
      052D 997 :
      052D 998 : RETURN TRUE RESULT
      052D 999 :
54 O1 D0 052D 1000 SETTRUE: MOVL #1,R4 ;SET RESULT TRUE
O5 05 0530 1002 RSB
      0531 1003
      0531 1004 .END

```

EXPRESS  
Symbol table

- EXPRESSION ANALYSIS

J 3

15-SEP-1984 23:46:42 VAX/VMS Macro V04-00  
4-SEP-1984 23:40:31 [DCL.SRC]EXPRESS.MAR;1

Page 28  
(17)

ADD	000004F1	R	02	OPI_K_GT	=	00000016		
AND	000004E2	R	02	OPI_K_GTS	=	0000002C		
CLIS_BUFOVF	= 00038018			OPI_K_LE	=	00000018		
CLIS_COMPLX	= 00038020			OPI_K_LES	=	0000002E		
CLIS_EXPSYN	= 00038038			OPI_K_LPAREN	=	00000022		
CLIS_IVCHAR	= 00038050			OPI_K_LT	=	0000001A		
CLIS_IVOPER	= 00038068			OPI_K_LTS	=	00000030		
CLIS_NORMAL	= 00030001			OPI_K_MUL	=	0000000A		
CLIS_UNDSYM	= 00038140			OPI_K_NE	=	0000001C		
CONCAT	00000457	R	02	OPI_K_NEG	=	0000000E		
CURMODE	00000002			OPI_K_NEGS	=	00000032		
DCL\$BACKUPMOVE	*****	X	02	OPI_K_NOT	=	00000002		
DCL\$BINEXPR	00000091	RG	02	OPI_K_OPERAND	=	00000034		
DCL\$CNVASCBIN	*****	X	02	OPI_K_OR	=	00000004		
DCL\$COMPRESS	*****	X	02	OPI_K_POS	=	00000010		
DCL\$CVT_BINARY	00000097	RG	02	OPI_K_RPAREN	=	00000024		
DCL\$CVT_STRING	*****	X	02	OPI_K_SOS	=	0000001E		
DCL\$EXPRADIX	000000C3	RG	02	OPI_K_STACK	=	00000036		
DCL\$EXPRESS	000000C0	RG	02	OPI_K_STORE	=	00000026		
DCL\$GETOKEN	*****	X	02	OPI_K_STRING	=	00000034		
DCL\$LEXIF	*****	X	02	OPI_K_SUB	=	00000008		
DCL\$MARK	*****	X	02	OPP_K_ADD	=	00000008		
DCL\$MOVBTKN	*****	X	02	OPP_K_AND	=	00000005		
DCL\$MOVCHAR	*****	X	02	OPP_K_DIV	=	00000009		
DCL\$PUTCHAR	*****	X	02	OPP_K_EOS	=	00000001		
DCL\$SEARCH	*****	X	02	OPP_K_EQ	=	00000007		
DCL\$SETCHAR	*****	X	02	OPP_K_EQS	=	00000007		
DCL\$UPCASE	*****	X	02	OPP_K_GE	=	00000007		
DISPATCH	000004C2	R	02	OPP_K_GES	=	00000007		
DIV	000004FE	R	02	OPP_K_GT	=	00000007		
EQL	00000518	R	02	OPP_K_GTS	=	00000007		
EVALUATE	00000303	R	02	OPP_K_LE	=	00000007		
EXIT	000003E6	R	02	OPP_K_LES	=	00000007		
FALSE	00000085	R	02	OPP_K_LPAREN	=	00000002		
FETCH	000003F7	R	02	OPP_K_LT	=	00000007		
FIND_OPERATOR	0000024A	R	02	OPP_K_LTS	=	00000007		
GEQ	0000051C	R	02	OPP_K_MUL	=	00000009		
GTR	00000520	R	02	OPP_K_NE	=	00000007		
LEQ	00000524	R	02	OPP_K_NEG	=	0000000A		
LIB\$SCOPY_DXDX	0000008E	RG	02	OPP_K_NEGS	=	00000007		
LOCALSIZ	0000000C			OPP_K_NOT	=	00000006		
LSS	00000528	R	02	OPP_K_OR	=	00000004		
MUL	000004FA	R	02	OPP_K_POS	=	0000000A		
NEG	00000513	R	02	OPP_K_RPAREN	=	00000003		
NEQ	000004DE	R	02	OPP_K_SOS	=	00000000		
NESTLVL	00000000			OPP_K_STORE	=	00000000		
NOT	000004E9	R	02	OPP_K_SUB	=	00000008		
NUMERIC	0000041C	R	02	OPTAB	=	00000000	R	02
OPERAND	00000401	R	02	OR	=	000004ED	R	02
OPI_K_ADD	= 00000006			PARSESTKSIZ	=	00000084		
OPI_K_AND	= 00000000			PARSE_OPERAND	=	0000027C	R	02
OPI_K_DIV	= 0000000C			POS	=	00000517	R	02
OPI_K_EOS	= 00000020			PRC_B_CONTINUE	=	000000F3		
OPI_K_EQ	= 00000012			PRC_B_DEFRADIX	=	000000AE		
OPI_K_EQS	= 00000028			PRC_B_EXMDEPMOD	=	000000AD		
OPI_K_GE	= 00000014			PRC_B_EXMDEPWID	=	000000AC		
OPI_K_GES	= 0000002A			PRC_B_EXONLYL	=	0000012D		

EXPRESS  
Symbol table

- EXPRESSION ANALYSIS

K 3

15-SEP-1984 23:46:42 VAX/VMS Macro V04-00  
4-SEP-1984 23:40:31 [DCL.SRC]EXPRESS.MAR;1

Page 20  
(17)

```

PRC_B_FLAGS2          000000AF
PRC_B_IMGFLAG         00000078
PRC_B_OUTFLAGS        0000012C
PRC_B_PROMPTLEN       000000F0
PRC_C_LENGTH          00000534
PRC_G_COMMANDS        00000133
PRC_G_PROMPT          000000F4
PRC_K_DEC              = 00000001
PRC_K_LENGTH          00000534
PRC_L_CURRKEY         00000048
PRC_L_EXMDEPADR       000000A8
PRC_L_EXTARG          00000094
PRC_L_EXTBLK          0000008C
PRC_L_EXTCOD          0000009C
PRC_L_EXTHND          00000090
PRC_L_EXTPRM          00000098
PRC_L_IDFLNK          0000008C
PRC_L_IMGACTSTS       00000080
PRC_L_INDCLOCK        0000007C
PRC_L_INDEPTH         0000005C
PRC_L_INDFAB          0000001C
PRC_L_INDINPRAB       00000014
PRC_L_INDOUTRAB       00000018
PRC_L_INPRAB          00000008
PRC_L_LASTKEY         0000004C
PRC_L_LSTSTATUS       000000B0
PRC_L_ONCTLY          000000B8
PRC_L_ONERROR         0000006C
PRC_L_OUTOFBAND       000000B4
PRC_L_OUTRAB          0000000C
PRC_L_OUTRABCTX       00000118
PRC_L_PPFLIST         00000070
PRC_L_RECALLPTR       0000012F
PRC_L_RESTART         00000058
PRC_L_SAVAP           00000000
PRC_L_SAVFP           00000004
PRC_L_SEVERITY        00000050
PRC_L_SPWN            000000C0
PRC_L_STACKLM         000000A4
PRC_L_STACKPT         000000A0
PRC_L_STATUS          00000054
PRC_L_STS             00000084
PRC_L_STV             00000088
PRC_L_SYMBOL          00000060
PRC_L_THBX            00000074
PRC_L_TRMLIST         00000010
PRC_Q_ALLOCREG        00000020
PRC_Q_COMMAND         000000E0
PRC_Q_FLUSHTIME       000000D0
PRC_Q_GLOBAL          00000028
PRC_Q_IMAGENAME       000000D8
PRC_Q_KEYPAD          00000040
PRC_Q_LABEL           00000030
PRC_Q_LOCAL           00000038
PRC_Q_SAVEPRIV        000000E8
PRC_T_OUTDVI          0000011C
PRC_W_ASTIOSB         000000C6

```

```

PRC_W_ASTRETN         000000C8
PRC_W_ASTSTATUS       000000C4
PRC_W_ATTMBX          0000007A
PRC_W_FLAGS           00000068
PRC_W_INPCHAN         00000064
PRC_W_ONLEVEL         0000006A
PRC_W_OUTIFI          00000114
PRC_W_OUTISI          00000116
PRC_W_OUTMBXCHN       000000CA
PRC_W_OUTMBXREF       000000CE
PRC_W_OUTMBXSIZ       000000CC
PRC_W_PMPTCTRL        000000F1
PRC_W_WAITIOSB        00000066
REDUCE                0000048D R    02
REQMODE               00000001
RESULT                00000004
SETFALSE              0000052A R    02
SETTRUE               0000052D R    02
STK_K_LENGTH         00000008
STK_L_ADDR            00000004
STK_W_PREC            00000002
STK_W_SIZE            00000002
STK_W_TYPE            00000000
STRING                0000040F R    02
STRINGDISP            00000453 R R   02
SUB                   000004F5 R    02
TRIADSTKSIZ          = 0000016C
TRUE                  00000081 R    02
TRUSYM                0000008A R    02
WRK_B_CMDOPT          FFFFFFF3
WRK_B_MAXPARM         FFFFFFFD0
WRK_B_MINPARM         FFFFFFFD1
WRK_B_PARMCNT         FFFFFFFCE
WRK_B_PARMSUM         FFFFFFFCF
WRK_B_RECALLCNT       FFFFFFFC5
WRK_B_VALLEV          FFFFFFFC4
WRK_B_VERBTYP         FFFFFFFC2
WRK_C_CMDBUFSIZ      = 00000400
WRK_C_LENGTH          FFFFF486
WRK_G_BUFFER          FFFFF492
WRK_G_INPBUF          FFFFF896
WRK_G_RESULT          FFFFF9B6
WRK_K_LENGTH          FFFFF486
WRK_L_CHARPTR         FFFFF48E
WRK_L_DISALLOW        FFFFFFFE6
WRK_L_ERRORRTN        FFFFF9AE
WRK_L_EXPANDPTR       FFFFF486
WRK_L_IMAGE           FFFFFFFE2
WRK_L_MARKPTR         FFFFF48A
WRK_L_PAROUT          FFFFFFFD2
WRK_L_PMPTADDR        FFFFF9A2
WRK_L_PROMPTRN        FFFFF9A6
WRK_L_PROPTR          FFFFFFFC6
WRK_L_QUABLK          FFFFFFFCA
WRK_L_READRTN         FFFFF9AA
WRK_L_RECALLPTR       FFFFFFFEA
WRK_L_RSLND           FFFFFFFB6

```



```

WRK_L_RSLNXT      FFFFFFFBA
WRK_L_SAVAP       FFFFFFFF8
WRK_L_SAVFP       FFFFFFFFC
WRK_L_SAVSP       FFFFFFFF4
WRK_L_SIGNALRTN  FFFFFFFD6
WRK_L_SPECTRN    FFFFFFF9B2
WRK_L_TAB_VEC     FFFFFFFDE
WRK_L_VERB        FFFFFFFBE
WRK_M_STAR        = 00000020
WRK_V_COMMENT     = 0000000C
WRK_W_FLAGS       FFFFFFFF0
WRK_W_FLAGS2      FFFFFFFF2
WRK_W_IMGCHAN     FFFFFFFEE
WRK_W_PMPTLEN     FFFFFFF99E
  _SS_            = 00000000
    
```

-----  
! Psect synopsis !  
-----

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 ( 0.)	00 ( 0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
\$ABSS	FFFFFFFC ( 0.)	01 ( 1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
DCL\$ZCODE	00000531 ( 1329.)	02 ( 2.)	NOPIC USR CON REL LCL NOSHR EXE RD NOWRT NOVEC BYTE

-----  
! Performance indicators !  
-----

Phase	Page faults	CPU Time	Elapsed Time
Initialization	13	00:00:00.04	00:00:01.81
Command processing	101	00:00:00.71	00:00:10.05
Pass 1	238	00:00:08.66	00:00:20.92
Symbol table sort	0	00:00:00.98	00:00:02.72
Pass 2	196	00:00:02.55	00:00:09.47
Symbol table output	30	00:00:00.19	00:00:00.85
Psect synopsis output	1	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	579	00:00:13.15	00:00:55.84

The working set limit was 1200 pages.  
 44122 bytes (87 pages) of virtual memory were used to buffer the intermediate code.  
 There were 40 pages of symbol table space allocated to hold 509 non-local and 146 local symbols.  
 1004 source lines were read in Pass 1, producing 20 object records in Pass 2.  
 29 pages of virtual memory were used to define 16 macros.

↑-----↑  
! Macro library statistics !  
↑-----↑

Macro library name	Macros defined
-----	-----
-\$255\$DUA28:[SYSLIB]SYSBLDMLB.MLB;1	0
-\$255\$DUA28:[DCL.OBJ]DCL.MLB;1	6
-\$255\$DUA28:[SYS.OBJ]LIB.MLB;1	0
-\$255\$DUA28:[SYSLIB]STARLET.MLB;2	3
TOTALS (all libraries)	9

513 GETS were required to define 9 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:EXPRESS/OBJ=OBJ\$:EXPRESS MSRC\$:EXPRESS/UPDATE=(ENH\$:EXPRESS)+EXECML\$/LIB+LIB\$:DCL/LIB+SYSS\$LIBRARY:SYSBLDMLB/LIB

Terminal Window 1	Terminal Window 2	Terminal Window 3	Terminal Window 4	Terminal Window 5	Terminal Window 6	Terminal Window 7	Terminal Window 8	Terminal Window 9	Terminal Window 10
Terminal Window 11	Terminal Window 12	Terminal Window 13	Terminal Window 14	Terminal Window 15	Terminal Window 16	Terminal Window 17	Terminal Window 18	Terminal Window 19	Terminal Window 20
Terminal Window 21	Terminal Window 22	Terminal Window 23	Terminal Window 24	Terminal Window 25	Terminal Window 26	Terminal Window 27	Terminal Window 28	Terminal Window 29	Terminal Window 30
Terminal Window 31	Terminal Window 32	Terminal Window 33	Terminal Window 34	Terminal Window 35	Terminal Window 36	Terminal Window 37	Terminal Window 38	Terminal Window 39	Terminal Window 40
Terminal Window 41	Terminal Window 42	Terminal Window 43	Terminal Window 44	Terminal Window 45	Terminal Window 46	Terminal Window 47	Terminal Window 48	Terminal Window 49	Terminal Window 50
Terminal Window 51	Terminal Window 52	Terminal Window 53	Terminal Window 54	Terminal Window 55	Terminal Window 56	Terminal Window 57	Terminal Window 58	Terminal Window 59	Terminal Window 60
Terminal Window 61	Terminal Window 62	Terminal Window 63	Terminal Window 64	Terminal Window 65	Terminal Window 66	Terminal Window 67	Terminal Window 68	Terminal Window 69	Terminal Window 70
Terminal Window 71	Terminal Window 72	Terminal Window 73	Terminal Window 74	Terminal Window 75	Terminal Window 76	Terminal Window 77	Terminal Window 78	Terminal Window 79	Terminal Window 80
Terminal Window 81	Terminal Window 82	Terminal Window 83	Terminal Window 84	Terminal Window 85	Terminal Window 86	Terminal Window 87	Terminal Window 88	Terminal Window 89	Terminal Window 90
Terminal Window 91	Terminal Window 92	Terminal Window 93	Terminal Window 94	Terminal Window 95	Terminal Window 96	Terminal Window 97	Terminal Window 98	Terminal Window 99	Terminal Window 100