

```
MMM      MMM      AAAAAAAAAA      CCCCCCCCCCCC      RRRRRRRRRRRR      0000000000
MMM      MMM      AAAAAAAAAA      CCCCCCCCCCCC      RRRRRRRRRRRR      0000000000
MMM      MMM      AAAAAAAAAA      CCCCCCCCCCCC      RRRRRRRRRRRR      0000000000
MMMMMM    MMMMM    AAA          AAA    CCC          RRR          RRR    000          000
MMMMMM    MMMMM    AAA          AAA    CCC          RRR          RRR    000          000
MMMMMM    MMMMM    AAA          AAA    CCC          RRR          RRR    000          000
MMM      MMM      AAA          AAA    CCC          RRR          RRR    000          000
MMM      MMM      AAA          AAA    CCC          RRR          RRR    000          000
MMM      MMM      AAA          AAA    CCC          RRR          RRR    000          000
MMM      MMM      AAA          AAA    CCC          RRR          RRR    000          000
MMM      MMM      AAA          AAA    CCC          RRR          RRR    000          000
MMM      MMM      AAA          AAA    CCC          RRR          RRR    000          000
MMM      MMM      AAA          AAA    CCC          RRR          RRR    000          000
MMM      MMM      AAAAAAAAAA      CCC          RRR          RRR    000          000
MMM      MMM      AAAAAAAAAA      CCC          RRR          RRR    000          000
MMM      MMM      AAAAAAAAAA      CCC          RRR          RRR    000          000
MMM      MMM      AAAAAAAAAA      CCC          RRR          RRR    000          000
MMM      MMM      AAAAAAAAAA      CCC          RRR          RRR    000          000
MMM      MMM      AAAAAAAAAA      CCC          RRR          RRR    000          000
MMM      MMM      AAA          AAA    CCC          RRR          RRR    000          000
MMM      MMM      AAA          AAA    CCC          RRR          RRR    000          000
MMM      MMM      AAA          AAA    CCC          RRR          RRR    000          000
MMM      MMM      AAA          AAA    CCC          RRR          RRR    000          000
MMM      MMM      AAA          AAA    CCCCCCCCCCCC      RRR          RRR    0000000000
MMM      MMM      AAA          AAA    CCCCCCCCCCCC      RRR          RRR    0000000000
MMM      MMM      AAA          AAA    CCCCCCCCCCCC      RRR          RRR    0000000000
```

```

PPPPPPPP      AAAAAA      RRRRRRRR      SSSSSSSS      EEEEEEEEEE      RRRRRRRR
PPPPPPPP      AAAAAA      RRRRRRRR      SSSSSSSS      EEEEEEEEEE      RRRRRRRR
PP      PP      AA      AA      RR      RR      SS      EE      RR      RR
PP      PP      AA      AA      RR      RR      SS      EE      RR      RR
PP      PP      AA      AA      RR      RR      SS      EE      RR      RR
PP      PP      AA      AA      RR      RR      SS      EE      RR      RR
PPPPPPPP      AA      AA      RRRRRRRR      SSSSSS      EEEEEEEE      RRRRRRRR
PPPPPPPP      AA      AA      RRRRRRRR      SSSSSS      EEEEEEEE      RRRRRRRR
PP      AAAAAAAAAA      RR      RR      SS      EE      RR      RR
PP      AAAAAAAAAA      RR      RR      SS      EE      RR      RR
PP      AA      AA      RR      RR      SS      EE      RR      RR
PP      AA      AA      RR      RR      SSSSSSSS      EEEEEEEEEE      RR      RR
PP      AA      AA      RR      RR      SSSSSSSS      EEEEEEEEEE      RR      RR

```

```

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      II      SS
LLLLLLLLLL      IIIIII      SSSSSSSS
LLLLLLLLLL      IIIIII      SSSSSSSS

```

(2) 55
(3) 84
(4) 282

DECLARATIONS
MAC\$PARSE PARSE VAX-11 MACRO PROGRAM
MAC\$TOKEN GET NEXT LEXICAL TOKEN

```
0000 1      .TITLE  MAC$PARSER PARSER FOR VAX-11 MACRO
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
0000 29     ++
0000 30     FACILITY:      VAX MACRO ASSEMBLER OBJECT LIBRARY
0000 31
0000 32     ABSTRACT:
0000 33
0000 34     The VAX-11 MACRO assembler translates MACRO-32 source code into object
0000 35     modules for input to the VAX-11 LINKER.
0000 36
0000 37     ENVIRONMENT:  USER MODE
0000 38
0000 39     AUTHOR: Benn Schreiber, CREATION DATE: 28-AUG-78
0000 40
0000 41     MODIFIED BY:
0000 42
0000 43     V02.06  HJ0001      Herb Jacobs      14-Aug-1980
0000 44     Performance improvement to parse driver loop.
0000 45
0000 46     V01.05  RN0023      R. Newland      3-Nov-1979
0000 47     New message codes to get error messages from system
0000 48     message file.
0000 49
0000 50     V01.04  RN0005      R. Newland      27-Aug-1979
0000 51     Remove .ALIGN LONG statements and change L^ to W^
0000 52
0000 53     --
```

```
0000 55 .SBTTL DECLARATIONS
0000 56 :
0000 57 : INCLUDE FILES:
0000 58 :
0000 59 :
0000 60 :
0000 61 : MACROS:
0000 62 :
0000 63 :
0000 64 $MAC_GENVALDEF ;DEFINE COMMON SYMBOLS
0000 65 $MAC_CTLFLGDEF ;DEFINE BIT FLAGS
0000 66 $MAC_INTCODEDEF ;DEFINE INT. FILE ACTIONS
0000 67 $MACMSGDEF ; Define message codes
0000 68
0000 69 :
0000 70 : EQUATED SYMBOLS:
0000 71 :
0000 72 :
FFFFFDD 0000 73 ELSE_CODE = -3 ;ELSE CODE
FFFFFDC 0000 74 CONT_CODE = -4 ;CONTINUE CODE
0000270E 0000 75 SCAN_CODE = 9998. ;SCAN CODE
0000270F 0000 76 ERR_CODE = 9999. ;ERROR CODE
00000009 0000 77 ERR_MAX = 9. ;MAX ERROR CODE
0000 78
0000 79 :
0000 80 : OWN STORAGE:
0000 81 :
0000 82
```

```
0000 84 .SBTTL MAC$PARSE PARSE VAX-11 MACRO PROGRAM
0000 85
0000 86 :++
0000 87 : FUNCTIONAL DESCRIPTION:
0000 88 : THE PARSE ROUTINE PERFORMS THE FOLLOWING ACTIONS:
0000 89 :
0000 90 : 1) INITIALIZE VARIABLES
0000 91 :
0000 92 : 2) MAKE AN INITIAL CALL TO 'MAC$GETCHR' TO GET THE FIRST
0000 93 : CHARACTER (AND FIRST DATA RECORD)
0000 94 :
0000 95 : 3) MAKE AN INITIAL CALL TO 'MAC$TOKEN' TO DETERMINE THE
0000 96 : FIRST LEXICAL ITEM, AND TO CLASSIFY IT INTO ONE OF
0000 97 : THE LEXICAL CLASS.
0000 98 :
0000 99 : 4) IT THEN DETERMINES THE NEXT TRANSITION STATE BY USING
0000 100 : THE CURRENT STATE AS <J> THE INDEX INTO THE STATE TABLE
0000 101 : OF TRANSITIONS.
0000 102 :
0000 103 : 5) THE JTH TRANSITION STATE IS THEN COMPARED TO THE CLASS
0000 104 : OF THE TOKEN. IF IT IS EQUAL TO THE <CLASS> OR IT IS
0000 105 : EQUAL TO THE <ELSE_CODE> THEN A 'MATCH' IS PERFORMED
0000 106 : AND THE SEQUENCE STARTS BACK AT 4).
0000 107 :
0000 108 : 6) IF THE <TRANSITION> DID NOT MATCH EITHER THE <ELSE_CODE>
0000 109 : OR THE <CLASS> THEN IF THE <TRANSITION> EQUALS THE
0000 110 : <CONT_CODE> THEN <J>, THE INDEX INTO THE <TRANSITION>
0000 111 : TABLE IS CHANGED TO BE THE JTH ITEM IN THE <ACTION>
0000 112 : TABLE. IF IT DOES NOT EQUAL THE CONTINUE CODE <J> IS
0000 113 : SIMPLY INCREMENTED BY 1.
0000 114 :
0000 115 : 7) ACTION NOW CONTINUES AT 5).
0000 116 :
0000 117 :
0000 118 : CALLING SEQUENCE:
0000 119 :
0000 120 : JSB MAC$PARSE
0000 121 :
0000 122 :
0000 123 : INPUT PARAMETERS:
0000 124 :
0000 125 : NONE
0000 126 :
0000 127 : IMPLICIT INPUTS:
0000 128 :
0000 129 : NONE
0000 130 :
0000 131 : OUTPUT PARAMETERS:
0000 132 :
0000 133 : NONE
0000 134 :
0000 135 : IMPLICIT OUTPUTS:
0000 136 :
0000 137 : NONE
0000 138 :
0000 139 : COMPLETION CODES:
0000 140 :
```

```
0000 141 : NONE
0000 142 :
0000 143 : SIDE EFFECTS:
0000 144 :
0000 145 : NONE
0000 146 :
0000 147 : REGISTER USAGE:
0000 148 :
0000 149 : R11 POINTER TO 'MAC$GL_FLAGS'
0000 150 : R10 CURRENT CHARACTER
0000 151 : R9 FRAME BUFFER POINTER
0000 152 : R8 TOKEN RETURNED FROM MAC$TOKEN AND SEMANTIC ROUTINES
0000 153 : R7 EXPRESSION STACK POINTER (0-100)
0000 154 : -----
0000 155 : REGISTERS R11-R7 MUST NOT BE ALTERED BY SEMANTIC ROUTINES.
0000 156 : REGISTERS R6-R0 ARE AVAILABLE FOR SEMANTIC ROUTINES TO USE
0000 157 : WITHOUT SAVING (SEE LRPTAB FOR A LIST OF SEMANTIC ROUTINES).
0000 158 : IN ADDITION, ANY ROUTINES CALLED BY 'MAC$TOKEN' MAY ALTER
0000 159 : REGISTERS R0-R6.
0000 160 : -----
0000 161 : R6 TOKEN IN PASS 1
0000 162 : R5 POINTER INTO PARSE TABLES
0000 163 : R4 CURRENT PARSE STATE
0000 164 : R3 ACTION
0000 165 : R2-R0 SCRATCH
0000 166 :
0000 167 : --
0000 168 :
0000 169 :
0000 170 :
00000000 171 : .PSECT MAC$RO_CODE_P1,NOWRT,GBL, LONG
0000 172 :
0000 173 : MAC$PARSE::
00 6B 01 E3 0000 174 : BBCCS #FLG$V_BOL,(R11),.+1 ;FLAG BEGINNING OF LINE
00 6B 0F E3 0004 175 : BBCCS #FLG$V_SKAN,(R11),.+1 ;FLAG SCANNING PERMITTED
00 6B 0D E5 0008 176 : BBCC #FLG$V_OPRND,(R11),.+1 ;NOT IN OPERAND FIELD
00 6B 20 E5 000C 177 : BBCC #FLG$V_CRSEEN,(R11),.+1 ;FLAG CR NOT SEEN YET FOR TOKEN
5A 0D 9A 0010 178 : MOVZBL #CR,R10 ;FORCE READING OF NEW LINE
FFEA' 30 0013 179 : BSBW MAC$GETCHR ;GET FIRST CHARACTER
57 D4 0016 180 : CLRL R7 ;INIT PARSE STACK POINTER
010F 30 0018 181 : BSBW MAC$TOKEN ;GET FIRST TOKEN TYPE
54 D4 001B 182 : CLRL R4 ;CLEAR CURRENT STATE
56 58 D0 001D 183 : MOVL R8,R6 ;SET CURRENT TOKEN CLASS
0020 184 : .ENABL LSB
0020 185 :
0020 186 : THIS CODE IS DEPENDENT ON THE SIZE OF THE ENTRIES IN THE SYMLST TABLE!
0020 187 :
0020 188 : PARSE_LOOP:
52 55 54 D0 0020 189 10$: MOVL R4,R5 ;COPY CURRENT STATE
00000000'EF45 9E 0023 190 15$: MOVAB L^PAT$AB_SYMLST[R5],R2 ;GET ADDRESS OF STATE TABLE OFFSET
56 B5 002B 191 : TSTW R6 ;ERROR PROCESSING TOKEN CLASS?
0D 12 002D 192 : BNEQ 20$ ;IF NE NO
62 09 91 002F 193 18$: CMPB #ERR_MAX,(R2) ;TOKEN CLASS ERROR?
2B 1E 0032 194 : BGEQU 40$ ;IF GTRU YES
82 FC 8F 91 0034 195 : CMPB #CONT_CODE,(R2)+ ;TOKEN CLASS EQUAL ELSE OR CONTINUE?
F5 1A 0038 196 : BGTRU 18$ ;BRANCH IF NEITHER TO STILL SEARCH
0B 11 003A 197 : BRB 25$ ;PROCESS ELSE OR CONTINUE
```

```

      56 62 91 003C 198
      1E 13 003F 199 20$: CMPB (R2),R6 ;STATE EQUAL TO TOKEN CLASS?
      82 FC 8F 91 0041 200 BEQL 40$ ;IF EQ YES
      F5 1A 0045 201 CMPB #CONT_CODE,(R2)+ ;TOKEN CLASS EQUAL ELSE OR CONTINUE?
      14 12 0047 202 BGTRU 20$ ;BRANCH IF NEITHER TO STILL SEARCH
53 00000001'EF 9E 0049 203 25$: BNEQ 41$ ;BRANCH IF ELSE
      52 53 C2 0050 204 MOVAB L^PAT$AB_SYMLST+1,R3 ;GET ADDR OF START OF TABLE (FIX R2+1)
55 00000000'EF 42 32 0053 205 SUBL R3,R2 ;FORM OFFSET INTO TABLE
      C6 11 005B 206 CVTWL L^PAT$AW_ACTION[R2],R5 ;GET CONTINUE LOCATION
      005D 207 BRB 15$
      005D 208 ;
      005D 209 ; PERFORM MATCH
      005D 210 ;
      005D 211 ; THIS CODE IS DEPENDENT ON THE SIZE OF THE ENTRIES IN THE SYMLST TABLE!
      005D 212 ;
53 00000000'EF 52 D7 005D 213 41$: DECL R2 ;ADJUST R2, WE WENT 1 TO FAR
      52 53 C2 005F 214 40$: MOVAB L^PAT$AB_SYMLST,R3 ;GET ADDRESS OF START OF TABLE
53 00000000'EF 42 32 0066 215 SUBL R3,R2 ;FORM OFFSET INTO TABLE
      53 270F 8F B1 0069 216 CVTWL L^PAT$AW_ACTION[R2],R3 ;GET ACTION TO PERFORM
      10 12 0071 217 CMPW #ERR_CODE,R3 ;ERROR DETECTED?
      56 B5 0076 218 BNEQ 60$ ;IF NEQ NO
      08 12 0078 219 TSTW R6 ;BACKTRACKING ERROR?
      57 D7 007A 220 BNEQ 50$ ;IF NEQ NO
54 0000'CF 47 D0 007C 221 DECL R7 ;YES--BACK UP PARSE STACK POINTER
      56 D4 007E 222 MOVL W^MAC$AL_PSTACK[R7],R4 ;Back up to previous state
      98 11 0084 223 50$: CLRL R6 ;START BACKTRACKING IF NOT ALREADY
53 D8F2 8F B1 0086 224 BRB 10$ ;
      0E 18 0088 225 60$: CMPW #-SCAN_CODE,R3 ;LOOK AHEAD (NO SCAN)?
      0000'CF 47 54 D0 008D 226 BGEQ 70$ ;IF GEQ YES
0000'CF 47 0000'CF D0 008F 227 MOVL R4,W^MAC$AL_PSTACK[R7] ;No--save current state
      53 B5 0095 228 MOVL W^MAC$GL_VALUE,W^MAC$AL_VALSTACK[R7] ;
      29 19 009D 229 70$: TSTW R3 ;TIME TO READ NEXT TOKEN?
      009F 230 BLSS 90$ ;IF LSS NO
      00A1 231 ;
      00A1 232 ; READ NEXT TOKEN
      00A1 233 ;
54 53 D0 00A1 234 MOVL R3,R4 ;SET CURRENT STATE TO ACTION
      57 D6 00A4 235 INCL R7 ;ADVANCE STACK POINTER
      OF 6B OF E2 00A6 236 BBSS #FLG$V_SKAN,(R11),80$ ;BR IF OK TO SCAN FOR NEXT TOKEN
      00AA 237 ; AND SET SCAN OK FLAG
      0000'CF D0 00AA 238 MOVL W^MAC$GL_VNEXT,- ;RESET CONTEXT,
      0000'CF 00AE 239 W^MAC$GL_VALUE ; symbol already scanned
56 0000'CF D0 00B1 240 MOVL W^MAC$GL_NEXT,R6 ; during a look-ahead
      FF67 31 00B6 241 BRW 10$ ;CONTINUE SCANNING
      0070 8F BB 00B9 242 80$: PUSHF #^M<R4,R5,R6> ;SAVE REGISTERS
      006A 30 00BD 243 BSBW MAC$TOKEN ;GET NEXT TOKEN
      0070 8F BA 00C0 244 POPR #^M<R4,R5,R6> ;RESTORE REGISTERS
      56 58 D0 00C4 245 MOVL R8,R6 ;SET CLASS TO TOKEN TYPE
      FF56 31 00C7 246 BRW 10$ ;CONTINUE SCANNING
      00CA 247 ;
      00CA 248 ; CALL SEMANTIC ROUTINE TO PERFORM REDUCTION TO NON-TERMINAL STATE
      00CA 249 ;
      DBF2 8F 53 B1 00CA 250 90$: CMPW R3,-SCAN_CODE ;NO-SCAN?
      17 14 00CF 251 BGTR 100$ ;IF GTR THEN NOT NO-SCAN
53 270E 8F A0 00D1 252 ADDW2 #SCAN_CODE,R3 ;NO-SCAN--CORRECT ACTION CODE
      57 D7 00D6 253 DECL R7 ;BACK UP STACK POINTER
      0000'CF 56 D0 00D8 254 MOVL R6,W^MAC$GL_NEXT ; Save current flags
```

```
0000'CF D0 00DD 255      MOVL  W^MAC$GL_VALUE,-
0000'CF      00E1 256      W^MAC$GL_VNEXT ; Save current value
00 6B 0F E5 00E4 257      BBCC  #FLGSV_SKAN,(R11),100$ ;PROHIBIT SCANNING
0000'CF 0000'CF47 D0 00E8 258 100$: MOVL  W^MAC$AL_VALSTACK[R7],-
      00F0 259      W^MAC$GL_VALUE ; Get current value
      53 53 CE 00F0 260      MNEGL R3,R3 ;GET ACTION ROUTINE NUMBER
52 00000000'EF43 D0 00F3 261      MOVL  L^PAT$AL_SEM[R3],R2 ;GET ADDRESS OF SEMANTIC ROUTINE
      0A 13 00FB 262      BEQL  110$ ;IF EQL NULL ACTION
      0078 8F BB 00FD 263      PUSHR #^M<R3,R4,R5,R6> ;SAVE REGISTERS
      62 16 0101 264 MAC$CALL SEM::
      0078 8F BA 0103 265      JSB  (R2) ;CALL SEMANTIC ROUTINE
52 00000000'E3 9A 0107 266      POPR  #^M<R3,R4,R5,R6> ;RESTORE REGISTERS
      57 52 C2 010E 267 110$: MOVZBL L^PAT$AB_POP(R3),R2 ;GET NUMBER OF ITEMS TO POP
54 0000'CF47 D0 0111 268      SUBL2 R2,R7 ;"POP" THE STACK
      0000'CF D0 0117 269      MOVL  W^MAC$AL_PSTACK[R7],R4 ; Get current state
      0000'CF47 D0 011B 270      MOVL  W^MAC$GL_VALUE,-
56 00000000'E3 9A 011F 271      W^MAC$AL_VALSTACK[R7] ; Put result on stack
      FEF7 31 0126 272      MOVZBL L^PAT$AB_LHS(R3),R6 ;PICK UP CLASS OF LEFT HAND SIDE
      0129 273      BRW  10$ ;CONTINUE SCANNING
      0129 274      ;
      0129 275      ; THE FOLLOWING ARE ROUTINES WHICH NEED TO BE DEFINED FOR THE
      0129 276      ; GRAMMAR. IT IS A NO-OP SEMANTIC ROUTINE.
      0129 277      ;
      0129 278 GOAL::
05 0129 279      RSB
      012A 280      .DSABL LSB
```

```
012A 282 .SBTTL MAC$TOKEN GET NEXT LEXICAL TOKEN
012A 283
012A 284 :++
012A 285 THE NEXT LEXICAL TOKEN TYPE IS RETURNED IN R8. THE VALUE
012A 286 ASSOCIATED WITH THE TOKEN IS RETURNED IN MAC$GL_VALUE.
012A 287 :--
012A 288
012A 289 MAC$TOKEN::
012A 290 MOVL W^MAC$GL_ERRPTX,-
012E 291 W^MAC$GL_ERRPT ;POINT TO PREVIOUS TOKEN
0131 292 MOVL W^MAC$GL_LINEPT,-
0135 293 W^MAC$GL_ERRPTX ;POINT TO THIS TOKEN
0138 294 CLRQ W^MAC$GL_VALUE ;DEFAULT VALUE IS 0
013C 295 10$: BSBW MAC$SKIPSP ;SKIP SPACES
013F 296 BBS #CHRSV_ILL_CHR,- ;BRANCH IF ILLEGAL CHR.
0141 297 W^MAC$AB_CMSK_TAB(R10),40$
0145 298 MOVL MAC$AL_CHRTAB[R10],R8 ;NO--GET TOKEN FOR CHARACTER
014D 299 BBCC #31.,R8,30$ ;BR IF NOT A TOKEN
0151 300 CMPB R10,#CR ;IS CHARACTER A CARRIAGE RETURN?
0154 301 BNEQ 20$ ;IF NEQ NO--GET NEXT CHARACTER
0156 302 BBSC #FLG$V_CRSEEN,(R11),20$ ;YES--HAVE WE SEEN IT BEFORE?
015A 303 ; (AND CLEAR FLAG)
015A 304 BBSC #FLG$V_CRSEEN,(R11),.+1 ;NO--SET FLAG FOR LATER
015E 305 RSB ;RETURN WITH CR TOKEN
015F 306 20$: BRW W^MAC$GETCHR ;GET NEXT CHARACTER
0162 307 ;AND RETURN TO CALLER
0162 308 :
0162 309 : CHARACTER IS NOT A TOKEN--CALL SCANNING ROUTINE
0162 310 :
0162 311 30$: JMP (R8) ;DISPATCH TO SCANNING ROUTINE
0164 312 :
0164 313 : CHARACTER IS ILLEGAL
0164 314 :
0164 315 40$:
0164 316 MAC$CHRRERR:: ;ENTRY FROM MAC$XUPARROW
0164 317 ; (MAC$XUPARROW IS JMP'ED TO
0164 318 ; FROM MAC$TOKEN)
0164 319 $MAC_ERR_ILLCHR ; Get message code
0169 320 BSSB MAC$ERRORLN ;REPORT CHARACTER ERROR
016B 321 BSBW MAC$GETCHR ;GET NEXT CHARACTER
016E 322 BRB MAC$TOKEN ;GET NEXT TOKEN
0170 323 :
0170 324 :++
0170 325 :
0170 326 : ERROR ROUTINES
0170 327 : ENTER WITH R0 CONTAINING THE ERROR MESSAGE INDEX.
0170 328 :--
0170 329 .ENABL LSB
0170 330
0170 331 MAC$ERRORPX:: ;ERROR USING MAC$GL_ERRPTX
0170 332 PUSHL W^MAC$GL_ERRPTX ;STACK POINTER
0174 333 BRB 10$
0176 334
0176 335 MAC$ERRORPT:: ;ERROR USING MAC$GL_ERRPT
0176 336 PUSHL W^MAC$GL_ERRPT ;STACK POINTER
017A 337 BRB 10$
017C 338
```

```
0000'CF DD 017C 339 MAC$ERRORLN::
00 6B 07 E5 017C 340 PUSHL W^MAC$GL_LINEPT ;ERROR USING MAC$GL_LINEPT
50 50 DD 0180 341 10$: BBCC #FLG$V_EXPOPT,(R11),.+1 ;STACK POINTER
50 12 9A 0184 342 PUSHL R0 ;DO NOT ALLOW EXPRESSION OPT.
FE74' 30 0186 343 MOVZBL #INT$ ERR,R0 ;STACK ERROR MESSAGE INDEX
05 0189 344 BSBW MAC$INTOUT_2_LW ;SET INT. CODE FOR ERROR
018C 345 ;OUTPUT TO INT. FILE
018C 346 ;(NOTE: MUST BE BSBW/RSB; SEE
018D 347 RSB ; INTOUT.MAR)
018D 348 ;ALL DONE
018D 349 .DSABL LSB
018D 350
018D 351 .END
```

MAC\$PARSER
Symbol table

PARSER FOR VAX-11 MACRO

D 7

16-SEP-1984 02:13:18
5-SEP-1984 01:49:44

VAX/VMS Macro V04-00
[MACRO.SRC]PARSER.MAR;1

Page 9
(4)

\$COUNT = 0000003B
ARG\$K_SIZE = 000003E8
AUD\$K_SIZE = 00000010
BLNK = 00000020
CHRS\$M_COMMA CR = 00000020
CHRS\$M_ILL_CHR = 00000040
CHRS\$M_NUM_BER = 00000010
CHRS\$M_SPA_MSK = 00000001
CHRS\$M_SYM_CH1 = 00000008
CHRS\$M_SYM_CHR = 00000004
CHRS\$M_SYM_DLM = 00000002
CHRS\$V_COMMA CR = 00000005
CHRS\$V_CVTLWC = 00000061
CHRS\$V_ILL_CHR = 00000006
CHRS\$V_NOCVT = 0000007F
CHRS\$V_NUM_BER = 00000004
CHRS\$V_SPA_MSK = 00000000
CHRS\$V_SYM_CH1 = 00000003
CHRS\$V_SYM_CHR = 00000002
CHRS\$V_SYM_DLM = 00000001
CONT_CODE = FFFFFFFC
CR = 0000000D
ELSE_CODE = FFFFFFFD
ERR_CODE = 0000270F
ERR_MAX = 00000009
FF = 0000000C
FLG\$M_ALLCHR = 00000001
FLG\$M_BOL = 00000002
FLG\$M_CHKLPND = 00100000
FLG\$M_COMPEXPR = 00000004
FLG\$M_CONT = 00000008
FLG\$M_CRF = 40000000
FLG\$M_CRSEEN = 00000001
FLG\$M_DATRPT = 00000010
FLG\$M_DBGOUT = 00004000
FLG\$M_DLMSTR = 00008000
FLG\$M_ENDMCH = 00000020
FLG\$M_EVALEXPR = 00000040
FLG\$M_EXPOPT = 00000080
FLG\$M_EXTERR = 00010000
FLG\$M_EXTWRN = 00020000
FLG\$M_FIRSTLN = 00000200
FLG\$M_IFSTAT = 00800000
FLG\$M_IIF = 00400000
FLG\$M_INSERT = 00000100
FLG\$M_IRPC = 20000000
FLG\$M_LEXOP = 00000002
FLG\$M_LSTXST = 00000200
FLG\$M_MAC2COL = 00000800
FLG\$M_MACL = 00000800
FLG\$M_MACLTB = 08000000
FLG\$M_MACTXT = 00010000
FLG\$M_MEBLST = 00001000
FLG\$M_MOREARG = 00002000
FLG\$M_MOREINP = 00000008
FLG\$M_NEWPND = 00000400
FLG\$M_NOREF = 01000000

FLG\$M_NTTYPEPC = 00000020
FLG\$M_NULCHR = 00040000
FLG\$M_OBJXST = 00200000
FLG\$M_OPNDCHK = 00000100
FLG\$M_OPRND = 00002000
FLG\$M_OPTVFLIDX = 00001000
FLG\$M_ORDLST = 00020000
FLG\$M_P2 = 00004000
FLG\$M_RPTIRP = 10000000
FLG\$M_SEQFIL = 02000000
FLG\$M_SKAN = 00008000
FLG\$M_SPECOP = 00000004
FLG\$M_SPLALL = 04000000
FLG\$M_STOIMF = 00040000
FLG\$M_SYM2COL = 00000400
FLG\$M_TOCLFG = 00080000
FLG\$M_UPAFLG = 00000010
FLG\$M_UPDFIL = 00000080
FLG\$M_UPMARG = 00000040
FLG\$M_XCRF = 80000000
FLG\$V_ALLCHR = 00000000
FLG\$V_BOL = 00000001
FLG\$V_CHKLPND = 00000014
FLG\$V_COMPEXPR = 00000002
FLG\$V_CONT = 00000003
FLG\$V_CRF = 0000001E
FLG\$V_CRSEEN = 00000020
FLG\$V_DATRPT = 00000004
FLG\$V_DBGOUT = 0000002E
FLG\$V_DLMSTR = 0000002F
FLG\$V_ENDMCH = 00000005
FLG\$V_EVALEXPR = 00000006
FLG\$V_EXPOPT = 00000007
FLG\$V_EXTERR = 00000030
FLG\$V_EXTWRN = 00000031
FLG\$V_FIRSTLN = 00000029
FLG\$V_IFSTAT = 00000017
FLG\$V_IIF = 00000016
FLG\$V_INSERT = 00000008
FLG\$V_IRPC = 0000001D
FLG\$V_LEXOP = 00000021
FLG\$V_LSTXST = 00000009
FLG\$V_MAC2COL = 0000002B
FLG\$V_MACL = 0000000B
FLG\$V_MACLTB = 0000001B
FLG\$V_MACTXT = 00000010
FLG\$V_MEBLST = 0000000C
FLG\$V_MOREARG = 0000002D
FLG\$V_MOREINP = 00000023
FLG\$V_NEWPND = 0000000A
FLG\$V_NOREF = 00000018
FLG\$V_NTTYPEPC = 00000025
FLG\$V_NULCHR = 00000032
FLG\$V_OBJXST = 00000015
FLG\$V_OPNDCHK = 00000028
FLG\$V_OPRND = 0000000D
FLG\$V_OPTVFLIDX = 0000002C

FLG\$V_ORDLST = 00000011
FLG\$V_P2 = 0000000E
FLG\$V_RPTIRP = 0000001C
FLG\$V_SEQFIL = 00000019
FLG\$V_SKAN = 0000000F
FLG\$V_SPECOP = 00000022
FLG\$V_SPLALL = 0000001A
FLG\$V_STOIMF = 00000012
FLG\$V_SYM2COL = 0000002A
FLG\$V_TOCLFG = 00000013
FLG\$V_UPAFLG = 00000024
FLG\$V_UPDFIL = 00000027
FLG\$V_UPMARG = 00000026
FLG\$V_XCRF = 0000001F
GOAL = 00000129 RG 02
HASHSZ = 0000007F
HYPHEN = 0000002D
INP\$K_BUFSIZ = 000003E8
INT\$K_BUFSIZ = 000013F4
INT\$K_BUFWRN = 00001390
INT\$_ADD = 00000001
INT\$_AND = 00000002
INT\$_ASH = 00000003
INT\$_ASN = 0000000C
INT\$_AUGPC = 0000000D
INT\$_BDST = 0000000E
INT\$_CHKL = 0000000F
INT\$_DIV = 00000004
INT\$_END = 00000010
INT\$_EPT = 00000011
INT\$_ERR = 00000012
INT\$_ETX = 00000013
INT\$_FNEWL = 00000014
INT\$_ILG = 00000000
INT\$_INFO = 0000003A
INT\$_LGLAB = 00000015
INT\$_MACL = 00000016
INT\$_MUL = 00000005
INT\$_NEG = 00000006
INT\$_NEWL = 00000017
INT\$_NEWP = 00000018
INT\$_NOT = 00000007
INT\$_OP = 00000019
INT\$_OR = 00000008
INT\$_PRIL = 0000001A
INT\$_PRT = 0000001B
INT\$_PSECT = 0000001C
INT\$_REDEF = 0000001D
INT\$_REF = 0000001E
INT\$_REST = 0000001F
INT\$_SAME = 00000009
INT\$_SAVE = 00000020
INT\$_SBTTL = 00000021
INT\$_SETFLAG = 00000022
INT\$_SETLONG = 00000023
INT\$_SPIC = 00000024
INT\$_SPID = 00000025

MAC\$PARSER
Symbol table

PARSER FOR VAX-11 MACRO

E 7

16-SEP-1984 02:13:18
5-SEP-1984 01:49:44

VAX/VMS Macro V04-00
[MACRO.SRC]PARSER.MAR;1

Page 10
(4)

INT\$_STIB	=	00000026		
INT\$_STIL	=	00000028		
INT\$_STIW	=	00000027		
INT\$_STKEPT	=	00000029		
INT\$_STKG	=	0000002A		
INT\$_STKL	=	0000002B		
INT\$_STKPC	=	0000002C		
INT\$_STKS	=	0000002D		
INT\$_STOB	=	00000034		
INT\$_STOL	=	0000002E		
INT\$_STOW	=	00000035		
INT\$_STRB	=	0000002F		
INT\$_STRL	=	00000031		
INT\$_STRSB	=	00000032		
INT\$_STRSW	=	00000033		
INT\$_STRW	=	00000030		
INT\$_STSB	=	00000036		
INT\$_STSW	=	00000037		
INT\$_SUB	=	0000000A		
INT\$_SUME	=	00000039		
INT\$_WRN	=	00000038		
INT\$_XOR	=	0000000B		
LST\$K_BUF\$IZ	=	00000086		
LST\$K_L_P_PAGE	=	0000003C		
LST\$K_TITLE_SIZ	=	00000028		
MAC\$AB_CMSK_TAB	*****		X	02
MAC\$AL_CHRTAB	*****		X	02
MAC\$AL_PSTACK	*****		X	02
MAC\$AL_VALSTACK	*****		X	02
MAC\$CALL_SEM	00000101	RG		02
MAC\$CHRERR	00000164	RG		02
MAC\$ERRORLN	0000017C	RG		02
MAC\$ERRORPT	00000176	RG		02
MAC\$ERRORPX	00000170	RG		02
MAC\$GETCHR	*****		X	02
MAC\$GL_ERRPT	*****		X	02
MAC\$GL_ERRPTX	*****		X	02
MAC\$GL_LINEPT	*****		X	02
MAC\$GL_NEXT	*****		X	02
MAC\$GL_VALUE	*****		X	02
MAC\$GL_VNEXT	*****		X	02
MAC\$INTOUT_2_LW	*****		X	02
MAC\$PARSE	00000000	RG		02
MAC\$SKIPSP	*****		X	02
MAC\$TOKEN	0000012A	RG		02
MAC\$_ILLCHR	=	007D90C2		
MAC\$_SUBSYS	=	0000007D		
OBJ\$K_BUF\$IZ	=	00000200		
OPF\$M_LASTOPR	=	00002000		
OPF\$M_OPTEXP	=	00001000		
OPF\$V_LASTOPR	=	0000000D		
OPF\$V_OPTEXP	=	0000000C		
PARSE_LOOP	00000020	R		02
PAT\$AB_LHS	*****		X	02
PAT\$AB_POP	*****		X	02
PAT\$AB_SYMLST	*****		X	02
PAT\$AL_SEM	*****		X	02

PAT\$AW_ACTION	*****	X	02
RD\$XSV_BINARY	=	00000000	
RD\$XSV_DECIMAL	=	00000002	
RD\$XSV_DOUBLE	=	00000005	
RD\$XSV_FLOAT	=	00000004	
RD\$XSV_GFLOAT	=	00000006	
RD\$XSV_HEX	=	00000003	
RD\$XSV_HFLOAT	=	00000007	
RD\$XSV_OCTAL	=	00000001	
REG\$PC	=	0000000F	
SCAN_CODE	=	0000270E	
SEMI	=	0000003B	
STB\$K_PG_MISS	=	0000000A	
SYM\$K_MAXLEN	=	0000001F	
SYM\$K_TWOCOL	=	00000010	
TAB	=	00000009	
X1	=	00000033	
X2	=	00080000	

+-----+
! 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	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
MAC\$RO_CODE_P1	0000018D (397.)	02 (2.)	NOPIC USR CON REL GBL NOSHR EXE RD NOWRT NOVEC LONG

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

Phase	Page faults	CPU Time	Elapsed Time
Initialization	36	00:00:00.04	00:00:02.58
Command processing	127	00:00:00.36	00:00:02.61
Pass 1	179	00:00:02.36	00:00:10.50
Symbol table sort	0	00:00:00.29	00:00:01.23
Pass 2	77	00:00:00.66	00:00:03.79
Symbol table output	20	00:00:00.09	00:00:00.15
Psect synopsis output	1	00:00:00.03	00:00:00.17
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	442	00:00:03.83	00:00:21.03

The working set limit was 1350 pages.

21216 bytes (42 pages) of virtual memory were used to buffer the intermediate code.

There were 20 pages of symbol table space allocated to hold 325 non-local and 19 local symbols.

351 source lines were read in Pass 1, producing 15 object records in Pass 2.

9 pages of virtual memory were used to define 8 macros.

+-----+
! Macro library statistics !
+-----+

Macro library name	Macros defined
_\$255\$DUA28:[MACRO.OBJ]MACRO.MLB;1	5
_\$255\$DUA28:[SYSLIB]STARLET.MLB;2	3
TOTALS (all libraries)	8

352 GETS were required to define 8 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:PARSER/OBJ=OBJ\$:PARSER MSRC\$:PARSER/UPDATE=(ENH\$:PARSER)+LIB\$:MACRO/LIB

0227 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

