
MMM MMM AAAAAAAAA CCCCCCCCCCC RRRRRRRRRRR 000000000
MMM MMM AAAAAAAAA CCCCCCCCCCC RRRRRRRRRRR 000000000
MMM MMM AAAAAAAAA CCCCCCCCCCC RRRRRRRRRRR 000000000
MMMMMM MMMMMM AAA AAA CCC RRR RRR OOO OOO
MMMMMM MMMMMM AAA AAA CCC RRR RRR OOO OOO
MMMMMM MMMMMM AAA AAA CCC RRR RRR OOO OOO
MMM MMM MMM AAA AAA CCC RRR RRR OOO OOO
MMM MMM MMM AAA AAA CCC RRR RRR OOO OOO
MMM MMM MMM AAA AAA CCC RRR RRR OOO OOO
MMM MMM AAA AAA CCC RRRRRRRRRRR OOO OOO
MMM MMM AAA AAA CCC RRRRRRRRRRR OOO OOO
MMM MMM AAA AAA CCC RRRRRRRRRRR OOO OOO
MMM MMM AAAAAAAAAAAAAAA CCC RRR RRR OOO OOO
MMM MMM AAAAAAAAAAAAAAA CCC RRR RRR OOO OOO
MMM MMM AAAAAAAAAAAAAAA CCC RRR RRR OOO OOO
MMM MMM AAA AAA CCC RRR RRR OOO OOO
MMM MMM AAA AAA CCC RRR RRR OOO OOO
MMM MMM AAA AAA CCC RRR RRR OOO OOO
MMM MMM AAA AAA CCCCCCCCCCC RRR RRR 000000000
MMM MMM AAA AAA CCCCCCCCCCC RRR RRR 000000000
MMM MMM AAA AAA CCCCCCCCCCC RRR RRR 000000000
MMM MMM AAA AAA CCCCCCCCCCC RRR RRR 000000000

```
FFFFFFFFFF  LL            000000      AAAAAA    TTTTTTTTTT
FFFFFFFFFF  LL            000000      AAAAAA    TTTTTTTTTT
FF          LL          00      00  AA      AA      TT
FF          LL          00      00  AA      AA      TT
FF          LL          00      00  AA      AA      TT
FF          LL          00      00  AA      AA      TT
FFFFFFFF    LL          00      00  AA      AA      TT
FFFFFFFF    LL          00      00  AA      AA      TT
FF          LL          00      00  AAAAAAAAAA      TT
FF          LL          00      00  AAAAAAAAAA      TT
FF          LL          00      00  AA      AA      TT          ....
FF          LL          00      00  AA      AA      TT          ....
FF          LLLLLLLLLL    000000    AA      AA      TT          ....
FF          LLLLLLLLLL    000000    AA      AA      TT          ....

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

```
MA
Sy

KT
KV
KW
KW
KW
KX
L
LS
LS
LS
MA
MA
MA
MA
MA
MA
MA
MA
MA
MA
MA
MA
MA
MA
MA
MA
MA
MA
MA
MA
MA
MA
MA
MB
MD
MF
MG
MH
ML
MO
MQ
MU
M-
NO
O
OB
OP
OP
OP
OP
OP
OP
OP
OP
OP
```

Table of contents

MA
Sy

PS
PS
PS
PS
PS
PS
PS
PS
PS
PS
PS
PS
PS
PS
PS
PS
Q
RB
RD
RD
RD
RD
RD
RD
RD
RD
RD
RD
RE
RF
RG
RH
RL
RL
RO
RQ
RR
RW
SE
SI
SI
ST
SY
SY
SY
SY
SY
SY
SY
SY
SY
SY
SY
SY
SY
SY
SY
SY
SY
SY
SY
SY
SY

```
0000     1          .TITLE  MAC$FLOAT       FLOATING POINT INPUT CONVERSION ROUTINE
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-11 MACRO assembler.
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  ;--
0000    38  ;
0000    39  ; VERSION: 01
0000    40  ;
0000    41  ; HISTORY:
0000    42  ;
0000    43  ; AUTHOR:
0000    44  ;       Peter Yuo, 25-Apr-77: Version 01
0000    45  ;
0000    46  ; MODIFIED BY:
0000    47  ;
0000    48  ;       V03-001 MTR0028         Mike Rhodes             4-Mar-1983
0000    49  ;                Change L^ references to shareable image to G^.
0000    50  ;
0000    51  ; 02-16 PCG0010         Peter George            Sep-08-1981
0000    52  ;       Push R7 at subroutine calls.
0000    53  ;
0000    54  ; 02-15 PCG0008         Peter George            Aug-27-1981
0000    55  ;       Rewrite all floating point routines.
0000    56  ;
0000    57  ; 01-4  Peter Yuo, 3-Jun-77
```

```
0000     58 ;
0000     59 ; 01-11 Benn Schreiber, 2-May-78
0000     60 ;
0000     61 ; 01-12 Benn Schreiber, 19-SEP-78, Implement rounding for MACRO32
0000     62 ;
0000     63 ; 01-13 R. Newland, 18-Oct-1979, Implement G_floating and H_floating support
0000     64 ;
0000     65 ; 01-14 R. Newland, 3-Nov-1979, New message codes
0000     66 ; 01-15 R. Newland, 13-Jan-1980, Trap floating oveflow when rounding floating
0000     67 ;                                point number.
0000     68 ;
0000     69
0000     70          .SBTTL  HISTORY         ; Detailed Current Edit History
0000     71
0000     72
0000     73 ; Edit History for Version 01 of FOR$FCNVIR
0000     74 ;
0000     75 ; 01-4  Add code to handle optional scale factor and digits in fraction
0000     76 ; 01-5  Fix bug in scale factor introduced in 01-4. Also shorten code
0000     77 ; 01-7  Fix bug in calculating S if there is a scale_factor
0000     78 ; 01-8  Fix bug in calculating S. If exponent field exists in input
0000     79 ;       P factor should be ignored.
0000     80 ; 01-9  Fixed bug in calculating S in order to take care of oveflow happened.
0000     81 ; 01-10 Change order of parameters to conform to standard.  JMT 15-Feb-78
0000     82 ; 01-11 Modify to work with VAX MACRO assembler
```

```
                     0000     84          .SBTTL  DECLARATIONS
                     0000     85
                     0000     86 ;
                     0000     87 ; INCLUDE FILES:
                     0000     88 ;
                     0000     89
                     0000     90 ;
                     0000     91 ; MACROS:
                     0000     92 ;
                     0000     93
                     0000     94          $MAC_SYMBLKDEF                  ; Define symbol block offsets
                     0000     95          $MAC_CTLFLGDEF                  ; Define control flags
                     0000     96          $MAC_GENVALDEF                  ; Define general values
                     0000     97          $MAC_GRAMMARDEF                 ; Define terminal grammar symbols
                     0000     98          $MAC_INTCODDEF                  ; Define int. buffer codes
                     0000     99          $MAC_INTCODDEF                  ; Define int. buffer codes
                     0000    100          $MAC_OPRDEF                     ; Define operand descriptor bits
                     0000    101          $MACMSGDEF                      ; Define message codes
                     0000    102          $SSDEF                          ; Define status codes
                     0000    103
                     0000    104 ;
                     0000    105 ; PSECT DECLARATIONS:
                     0000    106 ;
                     0000    107
                00000000     108          .PSECT  MAC$RO_DATA,NOEXE,NOWRT,GBL,LONG
                     0000    109 ;
                     0000    110 OTS_CVT:
00000000'            0000    111          .ADDRESS  OTS$CVT_T_F           ; Addresses of text to floating point
00000000'            0004    112          .ADDRESS  OTS$CVT_T_D           ; conversion routines
00000000'            0008    113          .ADDRESS  OTS$CVT_T_G
00000000'            000C    114          .ADDRESS  OTS$CVT_T_H
                     0010    115
                00000000     116          .PSECT  MAC$RO_CODE_P1,NOWRT,GBL,LONG
                     0000    117
                     0000    118 ;
                     0000    119 ; EQUATED SYMBOLS:
                     0000    120 ;
                     0000    121 ; The following symbols are used to indicate the bit position of the flag
                     0000    122 ; register.
                     0000    123 ;
                     0000    124
00000000             0000    125          V_DEC_POINT     = 0             ; Flag bit: 1 if decimal point is seen
00000001             0000    126          M_DEC_POINT     = ^X01          ; Mask for V_DEC_POINT
00000001             0000    127          V_EXP_LET       = 1             ; Flag bit: 1 if exponent is seen
00000002             0000    128          M_EXP_LET       = ^X02          ; Mask for V_EXP_LET
                     0000    129
                     0000    130 ;
                     0000    131 ; The following macro is used to store the current character in the temporary
                     0000    132 ; buffer, to increment the buffer length, and to get the next character.
                     0000    133 ;
                     0000    134
                     0000    135          .MACRO  GETCHR
                     0000    136          MOVB    R10,(R4)+               ; Move character to TMPBUF
                     0000    137          INCL    (R3)                    ; Increment string size
                     0000    138          BSBW    MAC$GETCHR              ; Get next character
                     0000    139          .ENDM   GETCHR
                     0000    140
```

```
                     0000   142          .SBTTL  MAC$READFLOAT           ; {D,E,F,G} format text reading routine
                     0000   143
                     0000   144 ;++
                     0000   145 ; FUNCTIONAL DESCRIPTION:
                     0000   146 ;
                     0000   147 ;       This routine copies floating point text from the input file,
                     0000   148 ;       into a temporary buffer.  It then calls the appropriate RTL
                     0000   149 ;       conversion routine to convert the text into the appropriate
                     0000   150 ;       internal floating point representation.
                     0000   151 ;
                     0000   152 ; INPUT PARAMETERS:
                     0000   153 ;
                     0000   154 ;       MAC$GL_CVTADDR  The address of the appropriate RTL routine to call.
                     0000   155 ;
                     0000   156 ; OUTPUT PARAMETERS:
                     0000   157 ;
                     0000   158 ;       MAC$GQ_VALUEQ   Contains the result.
                     0000   159 ;
                     0000   160 ; COMPLETION CODES:
                     0000   161 ;
                     0000   162 ;       R0              0       error in floating point syntax
                     0000   163 ;                       1       good floating point number
                     0000   164 ;
                     0000   165 ;--
                     0000   166
                     0000   167 MAC$READFLOAT::
                     0000   168
          55   D4    0000   169          CLRL    R5                      ; Clear flags
 53  0000'CF   9E    0002   170          MOVAB   W^MAC$AB_TMPBUF,R3      ; Set TMPBUF pointer
04 A3    08 A3   9E    0007   171          MOVAB   8(R3),4(R3)             ; and string descriptor pointer
          63   D4    000C   172          CLRL    (R3)                    ; Initialize string size to zero
    54    08 A3   9E    000E   173          MOVAB   8(R3),R4                ; and set output pointer
                     0012   174
       2D   5A   91  0012   175 SIGN:   CMPB    R10,#^A/-/              ; Is current char a "-" sign?
            05   13  0015   176          BEQL    SIGN_CONT               ; Branch if yes
       2B   5A   91  0017   177          CMPB    R10,#^A/+/              ; Is current char a "+" sign?
            08   12  001A   178          BNEQ    DEC_PT                  ; No, branch to decimal point test
                     001C   179 SIGN_CONT:
                     001C   180          GETCHR                          ; Yes, get next character
                     0024   181
                     0024   182 DEC_PT:
       2E   5A   91  0024   183          CMPB    R10,#^A/./              ; Is current char a "."?
            0B   12  0027   184          BNEQ    DIGIT_LOOP              ; No, branch to check if it is a digit
       55   01   88  0029   185          BISB    #M_DEC_POINT,R5         ; Set decimal point encounted flag
                     002C   186          GETCHR                          ; Get next character
                     0034   187
                     0034   188 DIGIT_LOOP:
       56   5A   9A  0034   189          MOVZBL  R10,R6                  ; Copy char for destruction
       56   30   C2  0037   190          SUBL    #^A/0/,R6               ; R6 = ASCII(current_char) - ASCII("0")
            0F   19  003A   191          BLSS    NOT_DIGIT               ; If lss then not a digit
       09   56   D1  003C   192          CMPL    R6,#9                   ; Check if current char is a digit
            0A   1A  003F   193          BGTRU   NOT_DIGIT               ; Branch if it is a digit
                     0041   194 DIGIT_CONT:
                     0041   195          GETCHR                          ; Get next character
            E9   11  0049   196          BRB     DIGIT_LOOP              ; Check if it is a digit
                     004B   197
                     004B   198 NOT_DIGIT:
```

```
                2E   5A   91   004B   199          CMPB    R10,#^A/./              ; Check if current char is a "."
                     06   12   004E   200          BNEQ    EXP_LET                 ; No, process exponent
           54 55     00   E2   0050   201          BBSS    #V_DEC_POINT,R5,ERROR   ; If second decimal point, then error
                     EB   11   0054   202          BRB     DIGIT_CONT              ; Get next digit
                                0056   203
                                0056   204 EXP_LET:
           32 55     01   E2   0056   205          BBSS    #V_EXP_LET,R5,CONVERT   ; If exponent already processed,
                                005A   206                                          ; Then finished reading number
              55     01   88   005A   207          BISB    #M_DEC_POINT,R5         ; Flag decimal point as seen
           44 8F     5A   91   005D   208          CMPB    R10,#^A/D/              ; "D"?
                     1E   13   0061   209          BEQL    EXP_SIGN                ; Process exponent sign
           45 8F     5A   91   0063   210          CMPB    R10,#^A/E/              ; "E"?
                     18   13   0067   211          BEQL    EXP_SIGN                ; Process exponent sign
           51 8F     5A   91   0069   212          CMPB    R10,#^A/Q/              ; "Q"?
                     12   13   006D   213          BEQL    EXP_SIGN                ; Process exponent sign
           64 8F     5A   91   006F   214          CMPB    R10,#^A/d/              ; "d"?
                     0C   13   0073   215          BEQL    EXP_SIGN                ; Process exponent sign
           65 8F     5A   91   0075   216          CMPB    R10,#^A/e/              ; "e"?
                     06   13   0079   217          BEQL    EXP_SIGN                ; Process exponent sign
           71 8F     5A   91   007B   218          CMPB    R10,#^A/q/              ; "q"?
                     0B   12   007F   219          BNEQ    CONVERT                 ; No exponent, so finished
                                0081   220
                                0081   221 EXP_SIGN:
                                0081   222          GETCHR                          ; Get next character
                   FF86   31   0089   223          BRW     SIGN                    ; Process sign character
                                008C   224
                                008C   225 CONVERT:
              0000'CF     7C   008C   226          CLRQ    W^MAC$GQ_VALUEQ         ; Clear value
              0000'CF     7C   0090   227          CLRQ    W^MAC$GQ_VAL2
                   01     DD   0094   228          PUSHL   #1                      ; Ignore spaces
                   00     DD   0096   229          PUSHL   #0
                   00     DD   0098   230          PUSHL   #0
              0000'CF     9F   009A   231          PUSHAB  W^MAC$GO_VALUEO         ; Address to put output
                   53     DD   009E   232          PUSHL   R3                      ; String descriptor address
        0000'DF    05     FB   00A0   233          CALLS   #5,@W^MAC$GL_CVTADDR    ; Call RTL conversion routine
                1A 50     E8   00A5   234          BLBS    R0,FINI                 ; OK if LBS
                                00A8   235
                                00A8   236 ERROR:
              0000'CF     7C   00A8   237          CLRQ    W^MAC$GQ_VALUEQ         ; Set result to zero
              0000'CF     7C   00AC   238          CLRQ    W^MAC$GQ_VAL2
                                00B0   239          $INTOUT_LW INT$_ERR,<#MAC$_FLTPNTSYNX,W^MAC$GL_LINEPT> ; Issue error
                   50     D4   00C0   240          CLRL    R0                      ; Flag an error
                                00C2   241
                                00C2   242 FINI:
              58   22     9A   00C2   243          MOVZBL  #DINTEGER,R8            ; Return token is DINTEGER
                          05   00C5   244          RSB
```

```
                     00C6   246          .SBTTL  FLOATING POINT LITERALS
                     00C6   247
                     00C6   248 ;
                     00C6   249 ;       These routines are called to process floating point literals.
                     00C6   250 ;
                     00C6   251
                     00C6   252 MAC$GETFLOAT::
00000000'GF    DE    00C6   253          MOVAL   G^OTS$CVT_T_F,-          ; Load address of RTL routine
     0000'CF          00CC   254                  W^MAC$GL_CVTADDR
         FF2E    30  00CF   255          BSBW    MAC$READFLOAT            ; Call input and conversion routine
                 05  00D2   256          RSB
                     00D3   257
                     00D3   258 MAC$GETDOUBLE::
00000000'GF    DE    00D3   259          MOVAL   G^OTS$CVT_T_D,-          ; Load address of RTL routine
     0000'CF          00D9   260                  W^MAC$GL_CVTADDR
         FF21    30  00DC   261          BSBW    MAC$READFLOAT            ; Call input and conversion routine
     0000'CF     D0  00DF   262          MOVL    W^MAC$GL_VAL3,-          ; Copy upper longword of value
     0000'CF          00E3   263                  W^MAC$GL_HIGH_32
                 05  00E6   264          RSB
                     00E7   265
                     00E7   266 MAC$GETGFLOAT::
00000000'GF    DE    00E7   267          MOVAL   G^OTS$CVT_T_G,-          ; Load address of RTL routine
     0000'CF          00ED   268                  W^MAC$GL_CVTADDR
         FF0D    30  00F0   269          BSBW    MAC$READFLOAT            ; Call input and conversion routine
     0000'CF     D0  00F3   270          MOVL    W^MAC$GL_VAL3,-          ; Copy upper longword of value
     0000'CF          00F7   271                  W^MAC$GL_HIGH_32
                 05  00FA   272          RSB
                     00FB   273
                     00FB   274 MAC$GETHFLOAT::
00000000'GF    DE    00FB   275          MOVAL   G^OTS$CVT_T_H,-          ; Load address of RTL routine
     0000'CF          0101   276                  W^MAC$GL_CVTADDR
         FEF9    30  0104   277          BSBW    MAC$READFLOAT            ; Call input and conversion routine
     0000'CF     D0  0107   278          MOVL    W^MAC$GL_VAL3,-          ; Copy upper longword of value
     0000'CF          010B   279                  W^MAC$GL_HIGH_32
     0000'CF     7D  010E   280          MOVQ    W^MAC$GQ_VAL2,-          ; Copy upper quadword of value
     0000'CF          0112   281                  W^MAC$GQ_HIGH_64
                 05  0115   282          RSB
                     0116   283
```

```
                     0116   285          .SBTTL  FLOATING POINT DIRECTIVES
                     0116   286
                     0116   287 ;
                     0116   288 ;       FLOAT, DOUBLE, GFLOAT, and HFLOAT are called to process
                     0116   289 ;       the .FLOAT, .DOUBLE, .G_FLOAT, and .H_FLOAT directives,
                     0116   290 ;       respectively.
                     0116   291 ;
                     0116   292
                     0116   293 FLOAT::                                        ;DATA_STAT = KFLOAT
     57    DD        0116   294          PUSHL   R7                           ; Save R7
57   04    D0        0118   295          MOVL    #RDX$V_FLOAT,R7              ; Indiate data type
     15    11        011B   296          BRB     GET_FLT_ARGS                 ; Join common code
                     011D   297
                     011D   298 DOUBLE::                                       ;DATA_STAT = KDOUBLE
     57    DD        011D   299          PUSHL   R7                           ; Save R7
57   05    D0        011F   300          MOVL    #RDX$V_DOUBLE,R7             ; Indiate data type
     0E    11        0122   301          BRB     GET_FLT_ARGS                 ; Join common code
                     0124   302
                     0124   303 GFLOAT::                                       ;DATA_STAT = KGFLOAT
     57    DD        0124   304          PUSHL   R7                           ; Save R7
57   06    D0        0126   305          MOVL    #RDX$V_GFLOAT,R7             ; Indiate data type
     07    11        0129   306          BRB     GET_FLT_ARGS                 ; Join common code
                     012B   307
                     012B   308 HFLOAT::                                       ;DATA_STAT = KHFLOAT
     57    DD        012B   309          PUSHL   R7                           ; Save R7
57   07    D0        012D   310          MOVL    #RDX$V_HFLOAT,R7             ; Indiate data type
     00    11        0130   311          BRB     GET_FLT_ARGS                 ; Join common code
                     0132   312
                     0132   313 ;
                     0132   314 ; Loop until the end-of-line, reading floating point numbers.  Emit
                     0132   315 ; code to pass 2 to stack the value, and if double precision, also
                     0132   316 ; stack the high order 32 bits.
                     0132   317 ;
                     0132   318
                     0132   319 GET_FLT_ARGS:
     57    04    CA   0132   320          BICL2   #4,R7                        ; Calculate RTL routine address
  57 57    02    78   0135   321          ASHL    #2,R7,R7
0000'CF 0000'C7  D0   0139   322          MOVL    W^OTS_CVT(R7),W^MAC$GL_CVTADDR
                     0140   323
        FEBD'  30     0140   324 10$:     BSBW    MAC$SKIPSP                   ; Skip any spaces
        FEBA   30     0143   325          BSBW    MAC$READFLOAT                ; Read floating point text
                     0146   326          $INTOUT_LW INT$_STIL,<W^MAC$GL_VALUE> ; Output bits 0-31
                     0150   327          $INC_PC #4                           ; Count them
          57   D5    0155   328          TSTL    R7                           ; Is it FLOAT?
          2D   13    0157   329          BEQL    20$                          ; Yes, then done outputing value
                     0159   330          $INTOUT_LW INT$_STIL,<W^MAC$GL_VAL3>  ; Output bits 32-63
                     0163   331          $INC_PC #4                           ; Count them
     0C   57    D1    0168   332          CMPL    R7,#<RDX$V_HFLOAT-RDX$V_FLOAT>*4  ; Is it HUGE?
          19    12    016B   333          BNEQ    20$                          ; No, then done outputing value
                     016D   334          $INTOUT_LW INT$_STIL,<W^MAC$GQ_VAL2+0> ; Output bits 64-95
                     0177   335          $INTOUT_LW INT$_STIL,<W^MAC$GQ_VAL2+4> ; Output bits 96-127
                     0181   336          $INC_PC #8                           ; Count them
                     0186   337
        FE77'  30     0186   338 20$:     BSBW    MAC$SKIPSP                   ; Skip spaces
     0D   5A    91    0189   339          CMPB    R10,#CR                      ; End of line
          1A    13    018C   340          BEQL    40$                          ; Yes if EQL
     2C   5A    91    018E   341          CMPB    R10,#^A/,/                   ; No--comma?
```

```
                10   13  0191   342          BEQL    30$                     ; if eql yes
                         0193   343          $MAC_ERR FLTPNTSYNX             ; No--get error code
           FE65'  30    0198   344          BSBW    MAC$ERRORLN             ; Issue error to pass 2
           FE62'  30    019B   345          BSBW    MAC$SKP_OPR             ; Skip to comma or eol
        0D   5A   91    019E   346          CMPB    R10,#CR                 ; Skip to end of line?
             05   13    01A1   347          BEQL    40$                     ; If eql yes--all done
           FE5A'  30    01A3   348 30$:      BSBW    MAC$GETCHR              ; Skip the comma
             98   11    01A6   349          BRB     10$                     ; Continue
        57   8E   D0    01A8   350 40$:      MOVL    (SP)+,R7                ; Restore R7
                  05    01AB   351          RSB                             ; All done
                        01AC   352
                        01AC   353          .END
```

```
$COUNT                  =  0000003B
AB                      =  00000001
AD                      =  0000C008
AF                      =  00008004
AG                      =  0000A008
AH                      =  00009010
AL                      =  00000004
AO                      =  00000010
AQ                      =  00000008
ARG$K_SIZE              =  000003E8
AUD$K_SIZE              =  00000010
AW                      =  00000002
B                       =  00000001
BLNK                    =  00000020
CHR$M_COMMA_CR          =  00000020
CHR$M_ILL_CHR           =  00000040
CHR$M_NUM_BER           =  00000010
CHR$M_SPA_MSK           =  00000001
CHR$M_SYM_CH1           =  00000008
CHR$M_SYM_CHR           =  00000004
CHR$M_SYM_DLM           =  00000002
CHR$V_COMMA_CR          =  00000005
CHR$V_CVTLWC            =  00000061
CHR$V_ILL_CHR           =  00000006
CHR$V_NOCVT             =  0000007F
CHR$V_NUM_BER           =  00000004
CHR$V_SPA_MSK           =  00000000
CHR$V_SYM_CH1           =  00000003
CHR$V_SYM_CHR           =  00000002
CHR$V_SYM_DLM           =  00000001
CNT                     =  00000001
CONVERT                    0000008C R      04
CR                      =  0000000D
D                       =  0000C008
DAND                    =  0000001D
DANGCLS                 =  00000016
DANGOPN                 =  00000015
DAT                     =  00000020
DBUP                    =  0000002B
DCLS                    =  00000018
DCOLON                  =  00000010
DCOMMA                  =  0000000F
DDIV                    =  0000001C
DEC_PT                     00000024 R      04
DEOL                    =  0000000B
DEQ                     =  00000011
DGUP                    =  0000002C
DIGIT_CONT                 00000041 R      04
DIGIT_LOOP                 00000034 R      04
DINTEGER                =  00000022
DIUP                    =  0000002D
DLUP                    =  0000002E
DMASK                   =  00000032
DMINUS                  =  0000001A
DOPCODE                 =  0000000E
DOPN                    =  00000017
DOR                     =  0000001E
DOUBLE                     0000011D RG     04
DPC                     =  00000012
DPLUS                   =  00000019
DPOUND                  =  00000021
DSQCLS                  =  00000014
DSQOPN                  =  00000013
DSUP                    =  0000002F
DTIMES                  =  0000001B
DUPA                    =  00000023
DUPB                    =  00000024
DUPC                    =  00000025
DUPD                    =  00000026
DUPF                    =  00000028
DUPM                    =  00000029
DUPO                    =  00000027
DUPX                    =  0000002A
DWUP                    =  00000030
DXOR                    =  0000001F
ERR                     =  00000000
ERR01                   =  00000001
ERR02                   =  00000002
ERR03                   =  00000003
ERR04                   =  00000004
ERR05                   =  00000005
ERR06                   =  00000006
ERR07                   =  00000007
ERR08                   =  00000008
ERR09                   =  00000009
ERROR                      000000A8 R      04
EXP_LET                    00000056 R      04
EXP_SIGN                   00000081 R      04
F                       =  00008004
FF                      =  0000000C
FINI                       000000C2 R      04
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$M_DLIMSTR           =  00008000
FLG$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$M_IIF               =  004000C0
FLG$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                    = 0000100C
FLG$M_MOREARG                   = 00002000
FLG$M_MOREINP                   = 00000008
FLG$M_NEWPND                    = 00000400
FLG$M_NOREF                     = 01000000
FLG$M_NTYPEPC                   = 00000020
FLG$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_TOCFLG                    = 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_DLIMSTR                   = 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_NTYPEPC                   = 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$V_TOCFLG                    = 00000013
FLG$V_UPAFLG                    = 00000024
FLG$V_UPDFIL                    = 00000027
FLG$V_UPMARG                    = 00000026
FLG$V_XCRF                      = 0000001F
FLOAT                             00000116 RG     04
G                               = 0000A008
GET_FLT_ARGS                      00000132 R      04
GFLOAT                            00000124 RG     04
GOALSY                          = 0000000A
H                               = 00009010
HASHSZ                          = 0000007F
HFLOAT                            0000012B RG     04
HYPHEN                          = 0000002D
ID                              = 0000000C
INP$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$_SPID               = 00000025
INT$_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
KADDRESS                = 00000037
KALIGN                  = 0000005A
KASCIC                  = 00000033
KASCID                  = 00000078
KASCII                  = 00000034
KASCIZ                  = 00000035
KBLKA                   = 0000003F
KBLKB                   = 00000040
KBLKD                   = 00000041
KBLKF                   = 00000042
KBLKG                   = 0000007E
KBLKH                   = 0000007F
KBLKL                   = 00000043
KBLKO                   = 00000080
KBLKQ                   = 00000044
KBLKW                   = 00000045
KBYTE                   = 00000038
KCROSS                  = 00000079
KDEBUG                  = 00000055
KDFLT                   = 0000007B
KDOUBLE                 = 00000039
KDSABL                  = 00000056
KENABL                  = 00000057
KEND                    = 00000076
KENDC                   = 0000004E
KENDM                   = 00000053
KENDR                   = 0000004F
KENTRY                  = 00000058
KERROR                  = 00000071
KEVEN                   = 0000005B
KEXTRN                  = 0000005D
KFIELD                  = 0000003A
KFLOAT                  = 0000003B
KGFLOAT                 = 00000081
KGLOBL                  = 0000005E
KHFLOAT                 = 00000082
KIDENT                  = 0000006A
KIF                     = 00000046
KIFF                    = 00000048
KIFT                    = 00000049
KIFTF                   = 0000004A
KIIF                    = 00000047
KINCLUDE                = 0000005F
KIRP                    = 0000004B
KIRPC                   = 0000004C
KLIBRARY                = 00000060
KLINK                   = 00000085
KLIST                   = 00000061
KLONG                   = 0000003C
KMACRO                  = 00000050
KMCALL                  = 00000051
KMDELETE                = 00000054
KMEXIT                  = 00000052
KNARG                   = 00000063
KNCHR                   = 00000064
KNCROS                  = 0000007A
KNLIST                  = 00000062
KNTYPE                  = 00000074
KOCTA                   = 00000083
KODD                    = 0000005C
KOPDEF                  = 00000075
KPACKED                 = 00000036
KPAGE                   = 00000065
KPRINT                  = 00000072
KPSECT                  = 00000066
KQUAD                   = 0000003D
KREF1                   = 0000006D
KREF16                  = 00000084
KREF2                   = 0000006E
KREF4                   = 0000006F
KREF8                   = 00000070
KREPT                   = 0000004D
KRESTORE                = 00000067
KSAVE                   = 00000068
KSBTTL                  = 0000006B
KSGNB                   = 0000007C
KSGNW                   = 0000007D
```

```
KTITLE                          = 00000069
KVECTOR                         = 00000059
KWARN                           = 00000073
KWEAK                           = 0000006C
KWORD                           = 0000003E
KXFER                           = 00000077
L                               = 00000004
LST$K_BUFSIZ                    = 00000086
LST$K_L_P_PAGE                  = 0000003C
LST$K_TITLE_SIZ                 = 00000028
MAC$AB_TMPBUF                     ********    X   04
MAC$ERRORLN                       ********    X   04
MAC$GETCHR                        ********    X   04
MAC$GETDOUBLE                     000000D3 RG     04
MAC$GETFLOAT                      000000C6 RG     04
MAC$GETGFLOAT                     000000E7 RG     04
MAC$GETHFLOAT                     000000FB RG     04
MAC$GL_CVTADDR                    ********      X 04
MAC$GL_HIGH_32                    ********      X 04
MAC$GL_LINEPT                     ********      X 04
MAC$GL_PC                         ********      X 04
MAC$GL_VAL3                       ********      X 04
MAC$GL_VALUE                      ********      X 04
MAC$GQ_VALUEQ                     ********      X 04
MAC$GQ_HIGH_64                    ********      X 04
MAC$GQ_VAL2                       ********      X 04
MAC$GQ_VALUEQ                     ********      X 04
MAC$INTERR_2_LW                   ********      X 04
MAC$INTOUT_1_LW                   ********      X 04
MAC$READFLOAT                     00000000 RG     04
MAC$SKIPSP                        ********    X   04
MAC$SKP_OPR                       ********    X   04
MAC$_FLTPNTSYNX                 = 007D9082
MACTRT                          = 0000000D
MAC_SUBSYS                      = 0000007D
MB                              = 00000041
MD                              = 0000C048
MF                              = 00008044
MG                              = 0000A048
MH                              = 00009050
ML                              = 00000044
MO                              = 00000050
MQ                              = 00000048
MW                              = 00000042
M_DEC_POINT                     = 00000001
M_EXP_LET                       = 00000002
NOT_DIGIT                         0000004B R      04
O                               = 00000010
OBJ$K_BUFSIZ                    = 00000200
OPD$M_ADDR                      = 00000000
OPD$M_BB                        = 000000A1
OPD$M_BW                        = 000000C2
OPD$M_D_FLOAT                   = 0000C000
OPD$M_F_FLOAT                   = 00008000
OPD$M_G_FLOAT                   = 0000A000
OPD$M_H_FLOAT                   = 00009000
OPD$M_MODE                      = 000003E0
OPD$M_MODIFY                    = 00000040
OPD$M_NOT_32F                   = 00007000
OPD$M_READ                      = 00000020
OPD$M_VIELD                     = 00000080
OPD$M_WRITE                     = 00000060
OPD$S_MODE                      = 00000005
OPD$S_SIZE                      = 00000005
OPD$V_D_FLOAT                   = 0000000E
OPD$V_F_FLOAT                   = 0000000F
OPD$V_G_FLOAT                   = 0000000D
OPD$V_H_FLOAT                   = 0000000C
OPD$V_MODE                      = 00000005
OPD$V_SIZE                      = 00000000
OPF$M_LASTOPR                   = 00002000
OPF$M_OPTEXP                    = 00001000
OPF$V_LASTOPR                   = 0000000D
OPF$V_OPTEXP                    = 0000000C
OTS$CVT_T_D                       ********      X 03
OTS$CVT_T_F                       ********      X 03
OTS$CVT_T_G                       ********      X 03
OTS$CVT_T_H                       ********      X 03
OTS_CVT                           00000000 R      03
PSC$B_NAME                        00000004
PSC$B_SEG                         0000000C
PSC$B_UNUSED                      0000000B
PSC$K_BLKSIZ                      00000013
PSC$K_NO_OPTNS                  = 0000000A
PSC$L_CURLOC                      0000000F
PSC$L_LINK                        00000000
PSC$L_MAXLGTH                     00000005
PSC$M_ABS                       = FFFFFFF7
PSC$M_ALIGNFLG                  = 00004000
PSC$M_ALLOPTNS                  = 000003FF
PSC$M_BYTE                      = 00004000
PSC$M_CON                       = FFFFFFFB
PSC$M_DEFAULT                   = 000001C8
PSC$M_EXE                       = 000000C0
PSC$M_GBL                       = 00000010
PSC$M_LCL                       = FFFFFFEF
PSC$M_LIB                       = 00000002
PSC$M_LONG                      = 00004800
PSC$M_NOEXE                     = FFFFFFBF
PSC$M_NOPIC                     = FFFFFFFE
PSC$M_NORD                      = FFFFFF7F
PSC$M_NOSHR                     = FFFFFFDF
PSC$M_NOVEC                     = FFFFFDFF
PSC$M_NOWRT                     = FFFFFEFF
PSC$M_OVR                       = 00000004
PSC$M_PAGE                      = 00006400
PSC$M_PIC                       = 00000001
PSC$M_QUAD                      = 00004C00
PSC$M_RD                        = 00000080
PSC$M_REL                       = 00000008
PSC$M_SHR                       = 00000020
PSC$M_USR                       = FFFFFFFD
PSC$M_VEC                       = 00000200
PSC$M_WORD                      = 00004400
```

```
PSC$M_WRT                       = 00000180
PSC$S_ALIGNMENT                 = 00000004
PSC$V_ALIGNFLG                  = 0000000E
PSC$V_ALIGNMENT                 = 0000000A
PSC$V_EXE                       = 00000006
PSC$V_GBL                       = 00000004
PSC$V_LIB                       = 00000001
PSC$V_OVR                       = 00000002
PSC$V_PIC                       = 00000000
PSC$V_RD                        = 00000007
PSC$V_REL                       = 00000003
PSC$V_SHR                       = 00000005
PSC$V_VEC                       = 00000009
PSC$V_WRT                       = 00000008
PSC$W_FLAG                        00000009
PSC$W_OPTIONS                     0000000D
Q                               = 00000008
RB                              = 00000021
RD                              = 0000C028
RDX$V_BINARY                    = 00000000
RDX$V_DECIMAL                   = 00000002
RDX$V_DOUBLE                    = 00000005
RDX$V_FLOAT                     = 00000004
RDX$V_GFLOAT                    = 00000006
RDX$V_HEX                       = 00000003
RDX$V_HFLOAT                    = 00000007
RDX$V_OCTAL                     = 00000001
REG$_PC                         = 0000000F
RF                              = 00008024
RG                              = 0000A028
RH                              = 00009030
RL                              = 00000024
RO                              = 00000030
RQ                              = 00000028
RRREG                           = 00000031
RW                              = 00000022
SEMI                            = 0000003B
SIGN                              00000012 R      04
SIGN_CONT                         0000001C R      04
STB$R_PG_MISS                   = 0000000A
SYM$B_NAME                        00000004
SYM$B_SEG                         0000000C
SYM$B_TOKEN                       0000000B
SYM$K_BLKSIZ                      0000000D
SYM$K_MAXLEN                    = 0000001F
SYM$K_TWOCOL                    = 00000010
SYM$L_LINK                        00000000
SYM$L_VAL                         00000005
SYM$M_ABS                       = 00000010
SYM$M_ASN                       = 00000100
SYM$M_CRFO                      = 00002000
SYM$M_DEBUG                     = 00000020
SYM$M_DEF                       = 00000001
SYM$M_DELMAC                    = 00000200
SYM$M_EPT                       = 00000200
SYM$M_EXTRN                     = 00000008
SYM$M_GLOBL                     = 00000004
SYM$M_LOCAL                     = 00000040
SYM$M_ODBG                      = 00000400
SYM$M_REF                       = 00000080
SYM$M_RELPSECT                  = 00000800
SYM$M_SUPR                      = 00004000
SYM$M_WEAK                      = 00000002
SYM$M_XCRF                      = 00001000
SYM$V_ABS                       = 00000004
SYM$V_ASN                       = 00000008
SYM$V_CRFO                      = 0000000D
SYM$V_DEBUG                     = 00000005
SYM$V_DEF                       = 00000000
SYM$V_DELMAC                    = 00000009
SYM$V_EPT                       = 00000009
SYM$V_EXTRN                     = 00000003
SYM$V_GLOBL                     = 00000002
SYM$V_LOCAL                     = 00000006
SYM$V_ODBG                      = 0000000A
SYM$V_REF                       = 00000007
SYM$V_RELPSECT                  = 0000000B
SYM$V_SUPR                      = 0000000E
SYM$V_WEAK                      = 00000001
SYM$V_XCRF                      = 0000000C
SYM$W_FLAG                        00000009
TAB                             = 00000009
VB                              = 00000081
VD                              = 0000C088
VF                              = 00008084
VG                              = 0000A088
VH                              = 00009090
VL                              = 00000084
VO                              = 00000090
VQ                              = 00000088
VW                              = 00000082
V_DEC_POINT                     = 00000000
V_EXP_LET                       = 00000001
W                               = 00000002
WB                              = 00000061
WD                              = 0000C068
WF                              = 00008064
WG                              = 0000A068
WH                              = 00009070
WL                              = 00000064
WO                              = 00000070
WQ                              = 00000068
WW                              = 00000062
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
. BLANK .	00000000 (0.)	01 (1.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
ABS	00000013 (19.)	02 (2.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
MAC$RO_DATA	00000010 (16.)	03 (3.)	NOPIC USR CON REL GBL NOSHR NOEXE RD NOWRT NOVEC LONG
MAC$RO_CODE_P1	000001AC (428.)	04 (4.)	NOPIC USR CON REL GBL NOSHR EXE RD NOWRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	31	00:00:00.05	00:00:00.92
Command processing	103	00:00:00.33	00:00:02.06
Pass 1	303	00:00:06.16	00:00:24.85
Symbol table sort	0	00:00:01.01	00:00:02.69
Pass 2	78	00:00:01.17	00:00:06.44
Symbol table output	64	00:00:00.32	00:00:02.16
Psect synopsis output	3	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	584	00:00:09.06	00:00:39.15

The working set limit was 1500 pages.
56202 bytes (110 pages) of virtual memory were used to buffer the intermediate code.
There were 60 pages of symbol table space allocated to hold 1047 non-local and 4 local symbols.
353 source lines were read in Pass 1, producing 19 object records in Pass 2.
19 pages of virtual memory were used to define 15 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[MACRO.OBJ]MACRO.MLB;1	13
_255DUA28:[SYSLIB]STARLET.MLB;2	4
TOTALS (all libraries)	17

1217 GETS were required to define 17 macros.

There were no errors, warnings or information messages.

```
MACRO/LIS=LIS$:FLOAT/OBJ=OBJ$:FLOAT MSRC$:FLOAT/UPDATE=(ENH$:FLOAT)+LIB$:MACRO/LIB
```

DATA
LIS

DEFINE
LIS

FLOAT
LIS

ERRMSG
LIS

GETARG
LIS

DATA1
LIS

INPUT
LIS

ERROR
LIS

FINISH
LIS

GETCMD
LIS