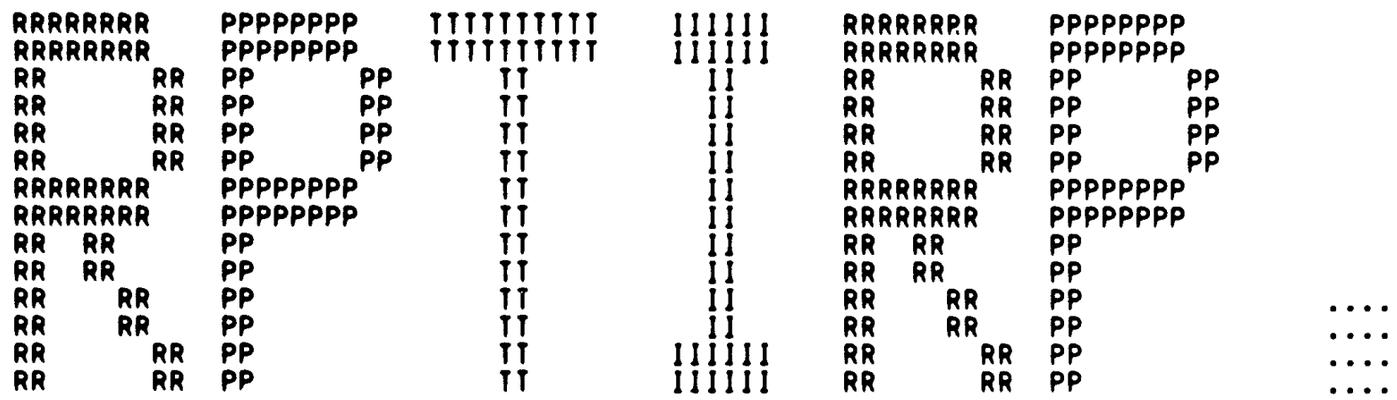

MMM MMM AAAAAAAAA CCCCCCCCCCC RRRRRRRRRRR 000000000
MMM MMM AAAAAAAAA CCCCCCCCCCC RRRRRRRRRRR 000000000
MMM MMM AAAAAAAAA CCCCCCCCCCC RRRRRRRRRRR 000000000
MMMMMM MMMMMM AAA AAA CCC RRR RRR OOO OOO
MMMMMM MMMMMM AAA AAA CCC RRR RRR OOO OOO
MMMMMM MMMMMM AAA AAA CCC RRR RRR OOO OOO
MMM MMM MMM AAA AAA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

FILEID**RPTIRP



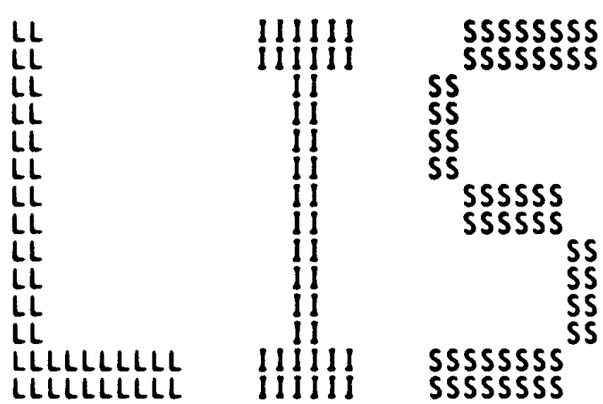


Table of contents

```
0000     1          .TITLE  MAC$RPTIRP      REPEAT AND INDEFINITE REPEAT PROCESSORS
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: 30-AUG-78
0000    40 ;
0000    41 ; MODIFIED BY:
0000    42 ;
0000    43 ;       V01.10  RN0023          R. Newland       3-Nov-1979
0000    44 ;               New message codes to get error message from system
0000    45 ;               message file.
0000    46 ;
0000    47 ;       V01.08  RN0010          R. Newland       5-Sep-1979
0000    48 ;               Multipage MXB blocks
0000    49 ;
0000    50 ;       V01.09  RN0012          R. Newland       26-Sep-1979
0000    51 ;               Fix problem with angle bracket processing of .IRPC
0000    52 ;               string argument.  SPR 11-25871
0000    53 ;
0000    54 ;       V01.07  RN0005          R. Newland       13-Aug-1979
0000    55 ;               Variable symbol storage
0000    56 ;
0000    57 ;--
```

```
                     59          .SBTTL  DECLARATIONS
                     60 ;
                     61 ; INCLUDE FILES:
                     62 ;
                     63
                     64 ;
                     65 ; MACROS:
                     66 ;
0000                 67          $MAC_CTLFLGDEF                  ;DEFINE CONTROL FLAGS
0000                 68          $MAC_GENVALDEF                  ;DEFINE GENERAL VALUES
0000                 69          $MAC_SYMBLKDEF                  ;DEFINE SYMBOL BLOCK OFFSETS
0000                 70          $MAC_MNBDEF                     ;DEFINE MACRO NAME BLOCK
0008                 71          $MAC_INPBLKDEF                  ;DEFINE INPUT BLOCK
003C                 72          $MAC_INTCODDEF                  ;DEFINE INT. BUFFER CODES
003C                 73          $MACMSGDEF                      ; Define message codes
003C                 74
003C                 75
003C                 76 ;
003C                 77 ; The following assumtions are made about the format of MNB and INP blocks
003C                 78 ;
003C                 79 ;
003C                 80          ASSUME  MNB$L_LINK      EQ      0
003C                 81          ASSUME  MNB$B_NAME      EQ      MNB$L_LINK+4
003C                 82          ASSUME  MNB$L_TXTP      EQ      MNB$B_NAME+1
003C                 83          ASSUME  MNB$W_FLAG      EQ      MNB$L_TXTP+4
003C                 84          ASSUME  MNB$L_PAGP      EQ      MNB$W_FLAG+2
003C                 85          ASSUME  MNB$L_PAGC      EQ      MNB$L_PAGP+4
003C                 86          ASSUME  MNB$L_CRSYMF    EQ      MNB$L_PAGC+4
003C                 87          ASSUME  MNB$B_ARGCT     EQ      MNB$L_CRSYMF+4
003C                 88          ASSUME  MNB$L_ARGP      EQ      MNB$B_ARGCT+1
003C                 89 ;
003C                 90 ;
003C                 91          ASSUME  INP$L_LINK      EQ      0
003C                 92          ASSUME  INP$L_NXTL      EQ      INP$L_LINK+4
003C                 93          ASSUME  INP$L_GETL      EQ      INP$L_NXTL+4
003C                 94          ASSUME  INP$L_IFLVL     EQ      INP$L_GETL+4
003C                 95          ASSUME  INP$L_IFVAL     EQ      INP$L_IFLVL+4
003C                 96          ASSUME  INP$L_RPTCNT    EQ      INP$L_IFVAL+4
003C                 97          ASSUME  INP$L_PAGP      EQ      INP$L_RPTCNT+4
003C                 98          ASSUME  INP$B_ARGCT     EQ      INP$L_PAGP+4
003C                 99          ASSUME  INP$L_ARGS      EQ      INP$B_ARGCT+1
003C                100 ;
003C                101 ;
003C                102 ;
00000000            103          .PSECT  MAC$RW_DATA, NOEXE, LONG
0000                104
0000                105
0000                106 BLOCK_SIZE:
00000004 0000       107          .BLKL   1                       ; Block size of MNB or MXB block
0004                108
00000000            109          .PSECT  MAC$RO_CODE_MAC,NOWRT,GBL,LONG
```

```
                        0000   111          .SBTTL  REPEAT DIRECTIVE PROCESSOR
                        0000   112
                        0000   113 ;++
                        0000   114 ; FUNCTIONAL DESCRIPTION:
                        0000   115 ;
                        0000   116 ;       THIS ROUTINE IS CALLED WHEN A .REPT DIRECTIVE IS SCANNED.
                        0000   117 ;       ALL THAT IS DONE IS TO CLEAR THE EVALUATE EXPRESSION FLAG
                        0000   118 ;       SO THAT NO CODE IS EMITTED TO PASS 2 TO EVALUATE EXPRESSIONS.
                        0000   119 ;
                        0000   120 ;--
                        0000   121
                        0000   122 REPTHD::                                     ;REPT_HEAD = KREPT
 00 6B   06     E5      000C   123          BBCC    #FLG$V_EVALEXPR,(R11),..+1 ;CLEAR EVALUATE EXPRESSION FLAG
    0000'CF     D4      0004   124          CLRL    W^MAC$GL_ABSFLAG           ;LOOK FOR ABSOLUTE EXPRESSION
                05      0008   125          RSB
                        0009   126
                        0009   127 ;++
                        0009   128 ; FUNCTIONAL DESCRIPTION:
                        0009   129 ;
                        0009   130 ;       THIS ROUTINE IS CALLED AFTER THE REPEAT COUNT OF A .REPT
                        0009   131 ;       DIRECTIVE HAS BEEN EVALUATED.  THE REPEAT BODY IS LOADED
                        0009   132 ;       INTO MEMORY, AND AN INPUT BLOCK IS CREATED TO READ THE
                        0009   133 ;       REPEAT BODY.
                        0009   134 ;
                        0009   135 ;--
                        0009   136
                        0009   137 ERREPT::                                    ;DIRECTIVE = REPT_HEAD ERR02
                        0009   138          $VPUSH  #0                         ;STACK A 0
          09    10      0011   139          BSBB    REPT                       ;PROCESS THE REPEAT
                        0013   140          $VPOP   R0                         ;CLEAR THE STACK
                05      001B   141          RSB
                        001C   142
                        001C   143 REPT::                                      ;DIRECTIVE = REPT_HEAD EXPR
          00 6B   06    E3     001C   144          BBCS    #FLG$V_EVALEXPR,(R11),..+1 ;RESET THE FLAG
                FFDD'   30     0020   145          BSBW    MAC$ALC_1_PAGE             ;ALLOCATE A PAGE
     0000'CF    50    D0      0023   146          MOVL    R0,W^MAC$GL_BLKPTR         ; Save address of block
                00000005      0028   147          .IF GREATER <INP$K_BLKSIZ-MNB$K_BLKSIZ>
     56   50    05    C1      0028   148          ADDL3   #<INP$K_BLKSIZ-MNB$K_BLKSIZ>,R0,R6 ; Allow for greater size of INP
                        002C   149          .IFF
                        002C   150          MOVL    R0,R6                      ; Copy the address
                        002C   151          .ENDC
66   1C   00    6E    00    2C     002C   152          MOVC5   #0,(SP),#0,#MNB$K_BLKSIZ,(R6) ; Zero the MACRO name block
                7E    D4     0032   153          CLRL    -(SP)                      ;SET NO MACRO ARGUMENTS
     50   53    0000'CF     C3     0034   154          SUBL3   W^MAC$GL_BLKPTR,R3,R0      ; Figure space used in block
          0000'CF    56     D0     003A   155          MOVL    R6,W^MAC$GL_MACPTR         ; Save address for BDYSCN
 7E   00000200 8F    50     C3     003F   156          SUBL3   R0,#512,-(SP)              ;FIGURE SPACE LEFT AND STACK
                53    DD     0047   157          PUSHL   R3                          ;STACK TEXT POINTER FOR CALLS
          00 6B   1C    E3     0049   158          BBCS    #FLG$V_RPTIRP,(R11),..+1 ;FLAG IN AN REPEAT/IRP
             5A   0D    D0     004D   159          MOVL    #CR,R10                  ;FORCE READING OF NEXT LINE
                FFAD'   30     0050   160          BSBW    MAC$GETCHR               ;READ THE FIRST CHAR OF BODY
          0000'CF    03    FB     0053   161          CALLS   #3,W^MAC$BODY_SCAN       ;SCAN THE REPEAT TEXT BODY
          00 6B   1C    E5     0058   162          BBCC    #FLG$V_RPTIRP,(R11),..+1 ;FLAG NOT IN REPEAT/IRP
                        005C   163 ;
                        005C   164 ; NOW WE WILL TURN THE MACRO NAME BLOCK AT THE BEGINNING OF THE
                        005C   165 ; BLOCK INTO AN INPUT BLOCK.
                        005C   166 ;
     50   0B A6    D0     005C   167          MOVL    MNB$L_PAGP(R6),R0          ;REMEMBER THE PAGE POINTER
```

```
                56   0000'CF    D0  0060   168          MOVL    W^MAC$GL_BLKPTR,R6      ; Get address of block
                66   0000'CF    D0  0065   169          MOVL    W^MAC$GL_INPLTP,(R6)    ;SET THE LINK
          0000'CF    86   DE    006A   170          MOVAL   (R6)+,W^MAC$GL_INPUTP   ;AND MAKE THIS THE CURRENT INPUT BLOCK
                51   0000'CF    D0  006F   171          MOVL    W^MAC$GL_ARGPTR,R1      ;GET POINTER TO WORD AFTER -1
                     86   71    3E  0074   172          MOVAW   -(R1),(R6)+             ;STORE POINTER TO END OF TEXT
                                     0077   173                                          ;SO THAT MAC$GET_RPT_LIN WILL
                                     0077   174                                          ;RESET
                86   0000'CF    9E  0077   175          MOVAB   W^MAC$GET_RPT_LIN,(R6)+ ;SET ROUTINE TO GET NEXT LINE
                51   0000'CF    9E  007C   176          MOVAB   W^MAC$GL_IF_LEVEL,R1    ;POINT TO IF LEVEL/VALUE
                     86   61    D0  0081   177          MOVL    (R1),(R6)+              ;SAVE OLD IF LEVEL
                          81    D4  0084   178          CLRL    (R1)+                   ;SET NEW IF LEVEL
                     86   61    D0  0086   179          MOVL    (R1),(R6)+              ;SAVE OLD IF VALUE
                          61    D4  0089   180          CLRL    (R1)                    ;SET NEW IF VALUE
                0000'CF    D5  008B   181          TSTL    W^MAC$GL_ABSFLAG        ;DID WE GET AN ABSOLUTE EXPRESSION?
                     10   13  008F   182          BEQL    40$                     ;IF EQL YES
                7E   50   7D  0091   183          MOVQ    R0,-(SP)                ;SAVE R0/R1
                                     0094   184          $MAC_ERR RPTCNTNABS             ; No--get message code
                   FF64'  30  0099   185          BSBW    MAC$ERRORPT             ;REPORT ERROR TO PASS 2
                50   8E   7D  009C   186          MOVQ    (SP)+,R0                ;RESTORE R0/R1
                     08   11  009F   187          BRB     50$                     ;AND USE 0
          51   0000'CF47   D0  00A1   188 40$:     MOVL    W^MAC$AL_VALSTACK[R7],R1;GET THE REPEAT COUNT
                     02   18  00A7   189          BGEQ    100$                    ;IF GEQ OK
                     51   D4  00A9   190 50$:     CLRL    R1                      ;ELSE REPEAT 0 TIMES
                86   51   D0  00AB   191 100$:    MOVL    R1,(R6)+                ;SET THE REPEAT COUNT
                86   50   D0  00AE   192          MOVL    R0,(R6)+                ;SET THE PAGE POINTER
                     86   94  00B1   193          CLRB    (R6)+                   ;NO ARGUMENTS
                     86   D4  00B3   194          CLRL    (R6)+                   ;CLEAR FIRST POINTER LONGWORD
                          05  00B5   195          RSB                             ;RETURN TO READ IN REPEAT TEXT
```

```
                                00B6   197          .SBTTL  INDEFINITE REPEAT DIRECTIVE PROCESSOR
                                00B6   198
                                00B6   199 ;++
                                00B6   200 ; FUNCTIONAL DESCRIPTION:
                                00B6   201 ;
                                00B6   202 ;       THIS ROUTINE SETS UP INDEFINTE REPEAT LOOPS.  A MACRO NAME BLOCK
                                00B6   203 ;       IS ALLOCATED AT THE BEGINNING OF A PAGE.  THE NAME IN THE MNB
                                00B6   204 ;       WILL BE THE NAME OF THE FORMAL ARGUMENT FOR THE REPEAT.  THE
                                00B6   205 ;       REAL ARGUMENTS ARE SCANNED AND INSERTED INTO A QUEUE.  THEN
                                00B6   206 ;       THE REPEAT BODY IS READ INTO CORE.  AFTER THE TEXT IS SCANNED,
                                00B6   207 ;       THE MACRO NAME BLOCK AT THE BEGINNING OF THE PAGE IS TRANSFORMED
                                00B6   208 ;       INTO AN INPUT CONTEXT BLOCK. THE INPUT CONTEXT IS THEN SWITCHED
                                00B6   209 ;       TO READ THE INDEFINTE REPEAT.
                                00B6   210 ;
                                00B6   211 ;--
                                00B6   212
                                00B6   213
                                00B6   214 IRPC::                                       ;DIRECTIVE = KIRPC
06 6B   1D   E3                 00B6   215          BBCS    #FLG$V_IRPC,(R11),IRP_0 ;FLAG THIS IS IRPC AND GO
        04   11                 00BA   216          BRB     IRP_0                   ;(IN CASE IT HAPPENED TO BE SET)
                                00BC   217
                                00BC   218 IRP::                                        ;DIRECTIVE = KIRP
00 6B   1D   E5                 00BC   219          BBCC    #FLG$V_IRPC,(R11),IRP_0 ;FLAG THIS IS IRP (NOT IRPC)
0E 0005'CF   E8                 00C0   220 IRP_0:   BLBS    W^LST$G_MACRODEF+SYM$[_VAL,10$ ; Branch if listing MACRO defs
                                00C5   221          $INTOUT_LW INT$_SETLONG,<#0,#MAC$GL_LIST_IT> ;NO--SEND TO PASS 2
      FF2A'  30                 00D3   222 10$:     BSBW    MAC$SYM$CNUP            ;SCAN FORMAL ARG NAME
   08 50     E8                 00D6   223          BLBS    R0,20$                  ;BRANCH IF WE FOUND ONE
                                00D9   224          $MAC_ERR NOFORMLARG             ; No--get message code
      FF1F'  31                 00DE   225          BRW     MAC$ERRORLN             ;ISSUE ERROR AND RETURN
1180 8F      BB                 00E1   226 20$:     PUSHR   #^M<R7,R8,R12>          ;SAVE REGISTERS
      FF18'  30                 00E5   227          BSBW    MAC$ALL_1_PAGE          ;ALLOCATE A PAGE
0000'CF 0200 8F   3C            00E8   228          MOVZWL  #512,W^BLOCK_SIZE       ; Set size of current block
        5C   50   D0            00EF   229          MOVL    R0,R12                  ;REMEMBER START OF CURRENT PAGE
   0000'CF   50   D0            00F2   230          MOVL    R0,W^MAC$GL_BLKPTR      ; Save block address
                                00F7   231 ;
                                00F7   232 ; NOW WE WILL SET UP AN MNB AT THE BEGINNING OF THE PAGE.  THE MACRO
                                00F7   233 ; NAME WILL BE THE FORMAL ARGUMENT NAME.
                                00F7   234 ;
    51  0C00'CF   9E            00F7   235          MOVAB   W^MAC$AB_TMPSYM,R1      ;POINT TO FORMAL ARG NAME
        52   61   9A            00FC   236          MOVZBL  (R1),R2                 ; Get length of name
             52   D6            00FF   237          INCL    R2                      ; and include count byte
    53  1C   52   C1            0101   238          ADDL3   R2,#MNB$K_BLKSIZ,R3     ; Get total bytes required for MNB
    53  3C   53   C3            0105   239          SUBL3   R3,#INP$K_IRPSIZ,R3     ; Is this less than size of INP block?
             03   13            0109   240          BEQL    25$                     ; No if EQL
        50   53   C0            010B   241          ADDL2   R3,R0                   ; Add size difference to MNB address
                                010E   242 25$:
        7E   52   90            010E   243          MOVB    R2,-(SP)
    60  61   52   28            0111   244          MOVC3   R2,(R1),(R0)            ; Copy formal arg name into MNB
        58   53   D0            0115   245          MOVL    R3,R8                   ; Save pointer
   0000'CF   58   D0            0118   246          MOVL    R8,W^MAC$GL_MACPTR
             83   D4            011D   247          CLRL    (R3)+                   ; Clear link word
        83   8E   90            011F   248          MOVB    (SP)+,(R3)+             ; Store offset to name
        83   01   B0            0122   249          MOVW    #1,(R3)+                ;STORE ARG # (OVER FIRST WORD OF
                                0125   250                                          ; THE TEXT POINTER)
             83   7C            0125   251          CLRQ    (R3)+                   ; Clear second word of text pointer,
                                0127   252                                          ; the flags word and page pointer
             83   7C            0127   253          CLRQ    (R3)+                   ; Clear PAGC and CRFSYM
```

```
                83    01    90  0129   254          MOVB    #1,(R3)+                ;ARG COUNT
                83    58    D0  012C   255          MOVL    R8,(R3)+                ;STORE POINTER TO ARGUMENT
          0000'CF     53    D0  012F   256          MOVL    R3,W^MAC$GL_ARGPTR      ;SAVE POINTER TO REAL ARG QUEUE
                63    63    DE  0134   257          MOVAL   (R3),(R3)               ;INIT THE REAL ARG QUEUE
                83    83    DE  0137   258          MOVAL   (R3)+,(R3)+             ;...
                                0013A   259                                          ;(NOTE: IF THIS IS IRPC THESE WORDS
                                0013A   260                                          ; WILL GET CHANGED LATER.)
                57    53    D0  0013A   261          MOVL    R3,R7                   ;SET POINTER INTO R7 FOR LATER
                   FEC0'    30  013D   262          BSBW    MAC$SKIPSP              ;SKIP SPACES
                2C    5A    91  0140   263          CMPB    R10,#^A/,/              ;STOP ON A COMMA?
                      06    12  0143   264          BNEQ    30$                     ;IF NEQ NO
                   FEB8'    30  0145   265          BSBW    MAC$GETCHR              ;YES--SKIP IT
                   FEB5'    30  0148   266          BSBW    MAC$SKIPSP              ;AND SKIP SPACES
             00 6B    05    E5  014B   267 30$:     BBCC    #FLG$V_ENDMCH,(R11),40$ ;FLAG NO LEFT ANGLE BRACKET SEEN
             0C 6B    1D    E0  014F   268 40$:     BBS     #FLG$V_IRPC,(R11),50$   ; If .IRPC let ARGSCN look after
                                0153   269                                          ; angle bracket processing
                3C    5A    91  0153   270          CMPB    R10,#^A/</              ; Left angle bracket?
                      07    12  0156   271          BNEQ    50$                     ;IF NEQ NO
                   FEA5'    30  0158   272          BSBW    MAC$GETCHR              ;YES--SKIP IT
             00 6B    05    E3  015B   273          BBCS    #FLG$V_ENDMCH,(R11),50$ ;FLAG WE SAW IT
                0D    5A    91  015F   274 50$:     CMPB    R10,#CR                 ;ARE WE AT EOL?
                      03    12  0162   275          BNEQ    SCAN_REAL_ARGS          ; If NEQ no--there are real arguments
                   008A    31  0164   276          BRW     NULL_REAL_ARGS          ; No actual arguments
                                0167   277
                                0167   278          .ENABL  LSB
                                0167   279
                                0167   280 ;
                                0167   281 ; NOW SCAN THE REAL ARGUMENTS, AND STORE THEM IN VIRTUAL MEMORY.
                                0167   282 ; THEY WILL BE LINKED TOGETER IN A QUEUE.  THE QUEUE HEADER IS
                                0167   283 ; LOCATED AT THE BEGINNING OF THE PAGE + MNB$K_BLKSIZ.  FOLLOWING
                                0167   284 ; THE QUEUE HEADER IS A LONGWORD POINTER TO THE INDEFINTE REPEAT
                                0167   285 ; TEXT.  THIS WILL BE FILLED IN LATER.
                                0167   286 ;
                                0167   287 SCAN_REAL_ARGS:
             00 6B    1C    E3  0167   288          BBCS    #FLG$V_RPTIRP,(R11),.+1 ;FLAG IN A REPEAT/IRP
                3E    5A    91  016B   289          CMPB    R10,#^A/>/              ;RIGHT ANGLE BRACKET?
                      13    12  016E   290          BNEQ    20$                     ;IF NEQ NO
                   FE8D'    30  0170   291          BSBW    MAC$GETCHR              ;YES--GET NEXT CHARACTER
                   FE8A'    30  0173   292          BSBW    MAC$SKIPSP              ;THEN SKIP SPACES
                0D    5A    91  0176   293          CMPB    R10,#CR                 ;STOP ON END OF LINE?
                      76    13  0179   294          BEQL    90$                     ;IF EQL YES--GO SCAN BODY
                                017B   295 10$:     $MAC_ERR DIRSYNX                ; No--get message code
                   FE7D'    30  0180   296          BSBW    MAC$ERRORLN             ;ISSUE MESSAGE TO PASS 2
                   FE7A'    30  0183   297 20$:     BSBW    MAC$MAC_ARG_SCN         ;SCAN THE REAL ARG
                56    50    D0  0186   298          MOVL    R0,R6                   ;REMEMBER THE LENGTH OF THE ARG
                      17    12  0189   299          BNEQ    40$                     ;IF NEQ OK
                0D    5A    91  018B   300          CMPB    R10,#CR                 ;NULL ARG--DID WE STOP ON EOL?
                      12    12  018E   301          BNEQ    40$                     ;IF NEQ NO--OK
             5D 6B    1D    E0  0190   302          BBS     #FLG$V_IRPC,(R11),90$   ; Branch if .IRPC
             78 6B    05    E1  0194   303          BBC     #FLG$V_ENDMCH,(R11),INGEST_BODY ;YES--BRANCH IF NO STARTING "<"
                                0198   304          $MAC_ERR DIRSYNX                ; Missing right angle bracket
                   FE60'    30  019D   305          BSBW    MAC$ERRORLN             ;ISSUE MESSAGE TO PASS 2
                      6E    11  01A0   306          BRB     INGEST_BODY             ;FINISH UP
          50    57    5C    C3  01A2   307 40$:     SUBL3   R12,R7,R0               ;FIGURE SPACE USED ON PAGE
       50 0000'CF    50    C3  01A6   308          SUBL3   R0,W^BLOCK_SIZE,R0      ; figure space left on block
                50    56    C2  01AC   309          SUBL2   R6,R0                   ;SEE IF ENOUGH ROOM FOR ARGUMENT
                50    0A    C2  01AF   310          SUBL2   #10,R0                  ; AND ARG LINK (2 LW) AND BYTE COUNT
```

```
                 1E   14  01B2   311          BGTR    50$                             ;BRANCH IF THERE IS ROOM
          51     56   12  C1  01B4   312          ADDL3   #<MXB$K_BLKSIZ+10>,R6,R1 ; Compute total size of block required
                    FE45'  30  01B8   313          BSBW    MAC$ALL_BLOCK           ; and allocate block of memory
    0000'CF       51   09  78  01BB   314          ASHL    #9,R1,W^BLOCK_SIZE      ; Set size of new block in bytes
                  5C   50  D0  01C1   315          MOVL    R0,R12                  ;SAVE PTR TO CURRENT PAGE
                           01C4   316          ASSUME  MXB$L_LINK  EQ 0
                           01C4   317          ASSUME  MXB$L_PAGES EQ MXB$L_LINK+4
            80     0B A8  D0  01C4   318          MOVL    MNB$L_PAGP(R8),(R0)+    ;LINK INTO PAGE LIST
                80   51  D0  01C8   319          MOVL    R1,(R0)+                ; Store block size in block
              0B A8   5C  D0  01CB   320          MOVL    R12,MNB$L_PAGP(R8)      ;...
                57   50  D0  01CF   321          MOVL    R0,R7                   ;UPDATE POINTER TO FREE SPOT
          50    0000'CF  D0  01D2   322 50$:     MOVL    W^MAC$GL_ARGPTR,R0      ;POINT TO THE ARGUMENT QUEUE HEAD
            21 6B   1D  E0  01D7   323          BBS     #FLG$V_IRPC,(R11),100$  ;BRANCH IF THIS IS IRPC
            04 B0   67  0E  01DB   324          INSQUE  (R7),@4(R0)             ;INSERT NEW ARG AT END QUEUE
                57   08  C0  01DF   325          ADDL2   #8,R7                   ;SKIP THE ARG QUEUE LINK WORDS
                87   56  B0  01E2   326          MOVW    R6,(R7)+                ;STORE LENGTH OF ARGUMENT
    67    0000'CF    56  28  01E5   327          MOVC3   R6,W^MAC$AB_TMPBUF,(R7) ;COPY ARG INTO PAGE
                57   53  D0  01EB   328          MOVL    R3,R7                   ;UPDATE PAGE POINTER
                   FF76  31  01EE   329          BRW     SCAN_REAL_ARGS
                         01F1   330 ;
                         01F1   331 ; THE REAL ARGUMENTS WERE NULL
                         01F1   332 ;
                         01F1   333 NULL_REAL_ARGS:
            1B 6B   1D  E1  01F1   334 90$:     BBC     #FLG$V_IRPC,(R11),INGEST_BODY ;BRANCH IF IRP (NOT IRPC)
          50    0000'CF  D0  01F5   335          MOVL    W^MAC$GL_ARGPTR,R0      ;IRPC--PICK UP QUEUE HEAD
                   56  D4  01FA   336          CLRL    R6                      ;CLEAR LENGTH OF STRING TO COPY
                         01FC   337 ;
                         01FC   338 ; THIS IS AN IRPC.  SET UP THE "ARGUMENT" FOR MAC$GET_IRC_LIN.
                         01FC   339 ; THEN COPY THE ARGUMENT STRING AND MARK THE END WITH A 0 BYTE.
                         01FC   340 ;
                80   01  B0  01FC   341 100$:    MOVW    #1,(R0)+                ;SET THE LENGTH OF THE ARGUMENT
                     80  94  01FF   342          CLRB    (R0)+                   ;CLEAR THE BYTE. (GET_IRC WILL
                         0201   343                                              ;FILL IT IN).
              80   04 A0  DE  0201   344          MOVAL   4(R0),(R0)+             ;SET UP THE POINTER INTO THE
                         0205   345                                              ;STRING TO IRPC OVER
    60    0000'CF    56  28  0205   346          MOVC3   R6,W^MAC$AB_TMPBUF,(R0) ;COPY THE IRPC STRING OUT
                     83  94  020B   347          CLRB    (R3)+                   ;MARK THE END WITH A ZERO BYTE
                57   53  D0  020D   348          MOVL    R3,R7                   ;UPDATE PAGE POINTER
                         0210   349          .DSABL  LSB
                         0210   350 ;
                         0210   351 ; HERE WHEN ALL ARGUMENTS HAVE BEEN SCANNED. NOW SCAN THE INDEFINTE
                         0210   352 ; REPEAT BODY AND STORE IT IN VIRTUAL MEMORY ALSO.
                         0210   353 ;
                         0210   354
                         0210   355 INGEST_BODY:
                58   DD  0210   356          PUSHL   R8                      ;POINT TO THE ONE ARGUMENT
          50    57   5C  C3  0212   357          SUBL3   R12,R7,R0               ;FIGURE SPACE USED
    7E    0000'CF    50  C3  0216   358          SUBL3   R0,W^BLOCK_SIZE,-(SP)   ; Figure space left
                06   6E  D1  021C   359          CMPL    (SP),#6                 ;IS THERE ENOUGH ROOM?
                     19  14  021F   360          BGTR    10$                     ;IF GTR YES
                   FDDC'  30  0221   361          BSBW    MAC$ALL_1_PAGE          ;NO--ALLOCATE A PAGE
                5C   50  D0  0224   362          MOVL    R0,R12                  ;SAVE POINTER TO CURRENT PAGE
            80     0B A8  D0  0227   363          MOVL    MNB$L_PAGP(R8),(R0)+    ;LINK INTO PAGE LIST
                80   01  D0  022B   364          MOVL    #1,(R0)+                ; Store block size in block
              0B A8   5C  D0  022E   365          MOVL    R12,MNB$L_PAGP(R8)      ;...
                57   50  D0  0232   366          MOVL    R0,R7                   ;UPDATE POINTER TO FREE SPOT
          6E    01F8 8F  3C  0235   367          MOVZWL  #<512-MXB$K_BLKSIZ>,(SP) ; Correct count on stack
```

```
                57   DD  023A   368 10$:     PUSHL   R7                      ;STACK POINTER FOR TEXT
             5A 0D   9A  023C   369          MOVZBL  #CR,R10                 ;FORCE READING OF NEW LINE
                FDBE' 30  023F   370          BSBW    MAC$GETCHR              ;READ FIRST CHARACTER OF THE BODY
        0000'CF 03   FB  0242   371          CALLS   #3,W^MAC$BODY_SCAN      ;SCAN THE REPEAT BODY
          00 6B 1C   E5  0247   372          BBCC    #FLG$V_RPTIRP,(R11),.+1 ;FLAG NOT IN REPEAT/IRP
                     024B   373 ;
                     024B   374 ; NOW TURN THE MNB INTO AN INPUT CONTEXT BLOCK AND SWITCH TO THE
                     024B   375 ; NEW CONTEXT BLOCK.
                     024B   376 ;
           50 0B A8  D0  024B   377          MOVL    MNB$L_PAGP(R8),R0       ;GET THE PAGE POINTER
        58 0000'CF   D0  024F   378          MOVL    W^MAC$GL_BLKPTR,R8      ; Get address for INP block
        68 0000'CF   D0  0254   379          MOVL    W^MAC$GL_INPUTP,(R8)    ;SET LINK INTO INPUT CONTEXT BLOCK
        56 0000'CF   D0  0259   380          MOVL    W^MAC$GL_MACPTR,R6      ; Get address of MNB
  0000'CF    88   DE  025E   381          MOVAL   (R8)+,W^MAC$GL_INPUTP   ;SET AS NEW INPUT CONTEXT BLOCK
        51 0000'CF   D0  0263   382          MOVL    W^MAC$GL_ARGPTR,R1      ;GET POINTER TO WORD AFTER -1
             88 71   3E  0268   383          MOVAW   -(R1),(R8)+             ;STORE PTR TO END OF TEXT SO
                     026B   384                                              ;THAT MAC$GET_IRP_LIN WILL GET
                     026B   385                                              ;FIRST REAL ARGUMENT
           07 6B 1D  E1  026B   386          BBC     #FLG$V_IRPC,(R11),20$   ;BRANCH IF THIS WAS IRP
        88 0000'CF   9E  026F   387          MOVAB   W^MAC$GET_IRC_LIN,(R8)+ ;NO--IRPC--SET INPUT LINE ROUTINE
                05   11  0274   388          BRB     30$                     ;CONTINUE
        88 0000'CF   9E  0276   389 20$:     MOVAB   W^MAC$GET_IRP_LIN,(R8)+ ;ADDRESS OF ROUTINE TO GET NEXT LINE
        51 0000'CF   9E  027B   390 30$:     MOVAB   W^MAC$GL_IF_LEVEL,R1    ;POINT TO IF LEVEL/VALUE
             88 61   D0  0280   391          MOVL    (R1),(R8)+              ;SAVE OLD IF LEVEL
                81   D4  0283   392          CLRL    (R1)+                   ;SET NEW IF LEVEL
             88 61   D0  0285   393          MOVL    (R1),(R8)+              ;SAVE OLD IF VALUE
                61   D4  0288   394          CLRL    (R1)                    ;CLEAR NEW IF VALUE
             88 57   D0  028A   395          MOVL    R7,(R8)+                ;STORE START OF TEXT POINTER
             88 50   D0  028D   396          MOVL    R0,(R8)+                ;STORE THE PAGE POINTER
             88 01   90  0290   397          MOVB    #1,(R8)+                ;ONE ARGUMENT
           06 6B 1D  E5  0293   398          BBCC    #FLG$V_IRPC,(R11),40$   ;BRANCH :  NOT IRPC (AND CLEAR FLAG)
           88 1C A6  DE  0297   399          MOVAL   MNB$K_BLKSIZ(R6),(R8)+  ;IRPC--POINT TO OUR "ARGUMENT"
                02   11  029B   400          BRB     50$                     ;CONTINUE
                88   D4  029D   401 40$:     CLRL    (R8)+                   ;CLEAR ARG POINTER--INPUT ROUTINE
                     029F   402                                              ;WILL SET IT LATER
                     029F   403 50$:
                     029F   404 IRP_EXIT:
              1180 8F    BA  029F   405          POPR    #^M<R7,R8,R12>          ;RESTORE REGISTERS
                     05   02A3   406          RSB                             ;RETURN TO READ FROM IRP TEXT
                     02A4   407
                     02A4   408          .END
```

```
$COUNT            = 0000003B
ARG$K_SIZE        = 000003E8
AUD$K_SIZE        = 00000010
BLNK              = 00000020
BLOCK_SIZE          00000000 R      03
CHR$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$V_SYM_DLM     = 00000001
CNT               = 00000002
CR                = 0000000D
ERR               = 00000000
ERREPT              00000009 RG     04
FF                = 0000000C
FLG$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$M_DLIMSTR     = 00008000
FLG$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$M_IIF         = 00400000
FLG$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$M_NTYPEPC     = 00000020
FLG$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$V_XCRF        = 0000001F
HASHSZ            = 0000007F
HYPHEN            = 0000002D
INGEST_BODY         00000210 R      04
INP$B_ARGCT         0000001C
INP$K_BLKSIZ        00000021
INP$K_BUFSIZ      = 000003E8
INP$K_IRPSIZ        0000003C
INP$L_ARGS          0000001D
INP$L_GETL          00000008
INP$L_IFLVL         0000000C
INP$L_IFVAL         00000010
INP$L_LINK          00000000
INP$L_NXTL          00000004
INP$L_PAGP          00000018
INP$L_RPTCNT        00000014
INT$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$_PSECT      = 0000001C          MAC$GET_IRP_LIN  ********  X  04    PSC$M_NORD      = FFFFFF7F
INT$_REDEF      = 0000001D          MAC$GET_RPT_LIN  ********  X  04    PSC$M_NOSHR     = FFFFFFDF
INT$_REF        = 0000001E          MAC$GL_ABSFLAG   ********  X  04    PSC$M_NOVEC     = FFFFFDFF
INT$_REST       = 0000001F          MAC$GL_ARGPTR    ********  X  04    PSC$M_NOWRT     = FFFFFEFF
INT$_SAME       = 00000009          MAC$GL_BLKPTR    ********  X  04    PSC$M_OVR       = 00000004
INT$_SAVE       = 00000020          MAC$GL_IF_LEVEL  ********  X  04    PSC$M_PAGE      = 00006400
INT$_SBTTL      = 00000021          MAC$GL_INPUTP    ********  X  04    PSC$M_PIC       = 00000001
INT$_SETFLAG    = 00000022          MAC$GL_LIST_IT   ********  X  04    PSC$M_QUAD      = 00004C00
INT$_SETLONG    = 00000023          MAC$GL_MACPTR    ********  X  04    PSC$M_RD        = 00000080
INT$_SPIC       = 00000024          MAC$INTOUT_2_LW  ********  X  04    PSC$M_REL       = 00000008
INT$_SPID       = 00000025          MAC$MAC_ARG_SCN  ********  X  04    PSC$M_SHR       = 00000C020
INT$_STIB       = 00000026          MAC$SKIPSP       ********  X  04    PSC$M_USR       = FFFFFFFD
INT$_STIL       = 00000028          MAC$SYMSCNUP     ********  X  04    PSC$M_VEC       = 00000200
INT$_STIW       = 00000027          MAC$_DIRSYNX     = 007D906A          PSC$M_WORD      = 00004400
INT$_STKEPT     = 0C000029          MAC$_NOFORMLARG= 007D9162            PSC$M_WRT       = 00000180
INT$_STKG       = 0000002A          MAC$_RPTCNTNABS= 007D91D2            PSC$S_ALIGNMENT= 00000004
INT$_STKL       = 0000002B          MAC_SUBSYS       = 0000007D          PSC$V_ALIGNFLG  = 0000000E
INT$_STKPC      = 0000002C          MNB$B_ARGCT      00000017            PSC$V_ALIGNMENT= 0000000A
INT$_STKS       = 0000002D          MNB$B_NAME       00000004            PSC$V_EXE       = 00000006
INT$_STOB       = 00000034          MNB$K_BLKSIZ     0000001C            PSC$V_GBL       = 00000004
INT$_STOL       = 0000002E          MNB$L_ARGP       00000018            PSC$V_LIB       = 00000001
INT$_STOW       = 00000035          MNB$L_CRSYMF     00000013            PSC$V_OVR       = 00000002
INT$_STRB       = 0000002F          MNB$L_LINK       00000000            PSC$V_PIC       = 00000000
INT$_STRL       = 00000031          MNB$L_PAGC       0000000F            PSC$V_RD        = 00000007
INT$_STRSB      = 00000032          MNB$L_PAGP       0000000B            PSC$V_REL       = 00000003
INT$_STRSW      = 00000033          MNB$L_TXTP       00000005            PSC$V_SHR       = 00000005
INT$_STRW       = 00000030          MNB$W_FLAG       00000009            PSC$V_VEC       = 00000009
INT$_STSB       = 00000036          MXB$K_BLKSIZ     00000008            PSC$V_WRT       = 00000008
INT$_STSW       = 00000037          MXB$L_LINK       00000000            PSC$W_FLAG        00000009
INT$_SUB        = 0000000A          MXB$L_PAGES      00000004            PSC$W_OPTIONS     0000000D
INT$_SUME       = 00000039          NULL_REAL_ARGS   000001F1 R  04      RDX$V_BINARY    = 00000000
INT$_WRN        = 00000038          OBJ$R_BUFSIZ     = 00000200          RDX$V_DECIMAL   = 00000002
INT$_XOR        = 0000000B          OPF$M_LASTOPR    = 00002000          RDX$V_DOUBLE    = 00000005
IRP               000000BC RG  04   OPF$M_OPTEXP     = 00001000          RDX$V_FLOAT     = 00000004
IRPC              000000B6 RG  04   OPF$V_LASTOPR    = 0000000D          RDX$V_GFLOAT    = 00000006
IRP_0             000000C0 R   04   OPF$V_OPTEXP     = 0000000C          RDX$V_HEX       = 00000003
IRP_EXIT          0000029F R   04   PSC$B_NAME        00000004          RDX$V_HFLOAT    = 00000007
LST$G_MACRODEF    ********    X  04   PSC$B_SEG         0000000C          RDX$V_OCTAL     = 00000001
LST$K_BUFSIZ    = 00000086            PSC$B_UNUSED      0000000B          REG$_PC         = 0000000F
LST$K_L_P_PAGE  = 0000003C            PSC$K_BLKSIZ      00000013          REPT              0000001C RG  04
LST$K_TITLE_SIZ= 00000028             PSC$K_NO_OPTNS  = 0000000A          REPTHD            00000000 RG  04
MAB$B_ARGNO       00000005            PSC$L_CURLOC      0000000F          SCAN_REAL_ARGS    00000167 R   04
MAB$B_NAME        00000004            PSC$L_LINK        00000000          SEMI            = 0000003B
MAB$K_BLKSIZ      0000000C            PSC$L_MAXLGTH     00000005          STB$K_PG_MISS   = 0000000A
MAB$L_DVPTR       00000008            PSC$M_ABS       = FFFFFFF7          SYM$B_NAME        00000004
MAB$L_LINK        00000000            PSC$M_ALIGNFLG  = 00004000          SYM$B_SEG         0000000C
MAB$W_DVLEN       00000006            PSC$M_ALLOPTNS  = 000003FF          SYM$B_TOKEN       0000000B
MAC$AB_TMPBUF   ********    X  04    PSC$M_BYTE      = 00004000          SYM$K_BLKSIZ      0000000D
MAC$AB_TMPSYM   ********    X  04    PSC$M_CON       = FFFFFFFB          SYM$K_MAXLEN    = 0000001F
MAC$ALL_1_PAGE  ********    X  04    PSC$M_DEFAULT   = 000001C8          SYM$K_TWOCOL    = 00000010
MAC$ALL_BLOCK   ********    X  04    PSC$M_EXE       = 000000C0          SYM$L_LINK        00000000
MAC$AL_VALSTACK ********    X  04    PSC$M_GBL       = 00000010          SYM$L_VAL         00000005
MAC$BODY_SCAN   ********    X  04    PSC$M_LCL       = FFFFFFEF          SYM$M_ABS       = 00000010
MAC$ERRORLN     ********    X  04    PSC$M_LIB       = 00000002          SYM$M_ASN       = 00000100
MAC$ERRORPT     ********    X  04    PSC$M_LONG      = 00004800          SYM$M_CRFO      = 00002000
MAC$GETCHR      ********    X  04    PSC$M_NOEXE     = FFFFFFBF          SYM$M_DEBUG     = 00000020
MAC$GET_IRC_LIN ********    X  04    PSC$M_NOPIC     = FFFFFFFE          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
X1              = 00000400
X2              = 0000000F
```

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	0000003C (60.)	02 (2.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
MAC$RW_DATA	00000004 (4.)	03 (3.)	NOPIC USR CON REL LCL NOSHR NOEXE RD WRT NOVEC LONG
MAC$RO_CODE_MAC	000002A4 (676.)	04 (4.)	NOPIC USR CON REL GBL NOSHR EXE RD NOWRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.04	00:00:02.81
Command processing	103	00:00:00.37	00:00:01.76
Pass 1	208	00:00:03.44	00:00:17.94
Symbol table sort	0	00:00:00.43	00:00:01.92
Pass 2	89	00:00:00.89	00:00:04.33
Symbol table output	31	00:00:00.16	00:00:00.78
Psect synopsis output	2	00:00:00.02	00:00:00.02

Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	464	00:00:05.35	00:00:29.56

The working set limit was 1350 pages.
31190 bytes (61 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 450 non-local and 20 local symbols.
408 source lines were read in Pass 1, producing 20 object records in Pass 2.
16 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

574 GETS were required to define 17 macros.

There were no errors, warnings or information messages.

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

MAIL

MAIL
MAP

P2DRVR
LIS

PARSER
LIS

RPTIRP
LIS

SCANER
LIS

TIMER
LIS

MAILCMDS
CLD

P2ACT2
LIS

SYMTAB
LIS