```
MMM          MMM     AAAAAAAAA         CCCCCCCCCCC   RRRRRRRRRRR        OOOOOOOOO
MMM          MMM     AAAAAAAAA         CCCCCCCCCCC   RRRRRRRRRRR        OOOOOOOOO
MMM          MMM     AAAAAAAAA         CCCCCCCCCCC   RRRRRRRRRRR        OOOOOOOOO
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      OOOOOOOOO
MMM         MMM    AAA         AAA      CCCCCCCCCCC   RRR         RRR      OOOOOOOOO
MMM         MMM    AAA         AAA      CCCCCCCCCCC   RRR         RRR      OOOOOOOOO
```

FILEID**ACTSTA

```
  AAAAAA      CCCCCCCC  TTTTTTTTTT    SSSSSSSS  TTTTTTTTTT    AAAAAA
  AAAAAA      CCCCCCCC  TTTTTTTTTT    SSSSSSSS  TTTTTTTTTT    AAAAAA
AA      AA  CC            TT      SS            TT      AA      AA
AA      AA  CC            TT      SS            TT      AA      AA
AA      AA  CC            TT      SS            TT      AA      AA
AA      AA  CC            TT      SS            TT      AA      AA
AA      AA  CC            TT        SSSSSS      TT      AA      AA
AA      AA  CC            TT        SSSSSS      TT      AA      AA
AAAAAAAAAA  CC            TT              SS    TT      AAAAAAAAAA
AAAAAAAAAA  CC            TT              SS    TT      AAAAAAAAAA
AA      AA  CC            TT              SS    TT      AA      AA      ....
AA      AA  CC            TT              SS    TT      AA      AA      ....
AA      AA    CCCCCCCC    TT      SSSSSSSS      TT      AA      AA      ....
AA      AA    CCCCCCCC    TT      SSSSSSSS      TT      AA      AA      ....

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
```

Table of contents

(2) 99 DECLARATIONS
(4) 189 OPCODE GENERATION
(5) 251 OPERAND GENERATION
(6) 396 ASSIGNMENT STATMENTS
(9) 578 BLOCK DATA STORAGE DIRECTIVES
(11) 665 LABEL DEFINITIONS
(12) 724 DATA GENERATION DIRECTIVES
(16) 938 ENTRY POINT DEFINITION DIRECTIVES

```
0000     1          .TITLE  MAC$ACTSTA MACHINE STATEMENTS
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: 25-AUG-78
0000    40 ;
0000    41 ; MODIFIED BY:
0000    42 ;
0000    43 ;       V03-002 MTR0034         Mike Rhodes     03-Jun-1983
0000    44 ;               Set SYM$M_REF in the current PSECT block when
0000    45 ;               a .MASK directive is encountered.
0000    46 ;
0000    47 ;       V03.01  MTR0017         Mike Rhodes     07-Jun-1982
0000    48 ;               Re-enable FLG$V_COMPEXPR in DATARG::, which
0000    49 ;               was diabled when a forward reference to a
0000    50 ;               symbol in an expression occurred.
0000    51 ;
0000    52 ;       V02.18  BLS0063         Benn Schreiber  30-Jul-1981
0000    53 ;               Remove 65K store repeated check since linker
0000    54 ;               allows more
0000    55 ;
0000    56 ;       V02.17  PCG0004         Peter George    28-Jul-1981
0000    57 ;               Call DATARG from QUDSTR and OCTSTR.
```

```
0000     58 ;
0000     59 ;       V02.16  PCG0002          Peter George    05-May-1981
0000     60 ;               Set RELPSECT flag for all global symbol assignments
0000     61 ;               and for all global labels.
0000     62 ;
0000     63 ;       V02.15  CNH0042          Chris Hume      28-Oct-1980
0000     64 ;               De-optimize boundary valued backward references if indexing
0000     65 ;               requested.  Allow the architecturally legal immediate mode in
0000     66 ;               address and vield contexts and also the practically useless
0000     67 ;               indexed immediate mode.
0000     68 ;               (ACTREF.MAR 02.15, DEFINE.MAR 02.17, SYMTAB.MAR 02.18)
0000     69 ;
0000     70 ;       V01.14  RN0023           R. Newland       3-Nov-1979
0000     71 ;               New message codes to get error messages from system
0000     72 ;               message file.
0000     73 ;
0000     74 ;       V01.13  RN0020           R. Newland      26-Oct-1979
0000     75 ;               Change error message for .BLKx expression not absolute
0000     76 ;
0000     77 ;       V01.12  RN0019           R. Newland      25-Oct-1979
0000     78 ;               Improve error pointer positioning
0000     79 ;
0000     80 ;       V01.11  RN0014           R. Newland      14-Oct-1979
0000     81 ;               Support for G_floating, H_floating and Octaword data types.
0000     82 ;               .BLKG, .BLKH, .BLKO and .OCTA directives.
0000     83 ;
0000     84 ;       V01.10  RN0005           R. Newland      12-Aug-1979
0000     85 ;               Remove .ALIGN LONG statements
0000     86 ;
0000     87 ;       V01.15  RN0029           R. Newland      12-Feb-1980
0000     88 ;               Correct listing of branch operand when on continued line.
0000     89 ;
0000     90 ;       V01.13  RN0021           R. Newland      28-Oct-1979
0000     91 ;               Correct listing of .ENTRY register mask value.
0000     92 ;               SPR 11-26384
0000     93 ;
0000     94 ;       V01.09  0003             B. Schreiber    10-JAN-1979
0000     95 ;               Catch syntax error if pound sign forgotten before
0000     96 ;               ASCII immediate (^A) in operands.
0000     97 ;--
```

```
                        0000     99          .SBTTL  DECLARATIONS
                        0000    100 ;
                        0000    101 ; INCLUDE FILES:
                        0000    102 ;
                        0000    103
                        0000    104 ;
                        0000    105 ; MACROS:
                        0000    106 ;
                        0000    107
                        0000    108          $MAC_SYMBLKDEF                  ;DEFINE SYMBOL BLOCK OFFSETS
                        0000    109          $MAC_CTLFLGDEF                  ;DEFINE CONTROL FLAGS
                        0000    110          $MAC_GENVALDEF                  ;DEFINE GENERAL VALUES
                        0000    111          $MAC_INTCODDEF                  ;DEFINE INT. FILE COMMANDS
                        0000    112          $MAC_ADRMODDEF                  ;DEFINE ADDRESSING MODES
                        0000    113          $MAC_OPRDEF                     ;DEFINE OPERAND DESCRIPTOR BITS
                        0000    114          $MACMSGDEF                      ; Define message codes
                        0000    115
                        0000    116 ;
                        0000    117 ; EQUATED SYMBOLS:
                        0000    118 ;
                        0000    119
                        0000    120 ;
                        0000    121 ; OWN STORAGE:
                        0000    122 ;
                        0000    123
                    00000000    124          .PSECT  MAC$RO_DATA,NOWRT,NOEXE,GBL,LONG
                        0000    125
                        0000    126 DAT_NUL_CMD:
                     26 0000    127          .BYTE   INT$_STIB,-
                        0001    128                  INT$_STIW,-
                     27 0001    129                  INT$_STIL,-
                     28 0002    130                  INT$_STIL,-
                     28 0003    131                  INT$_STIB,-
                     26 0004    132                  INT$_STIW,-
                  28 27 0005    133                  INT$_STIL
                        0007    134
                        0007    135 DAT_RPT_CMD:
                        0007    136          .BYTE   INT$_STRB,-
                     2F 0007    137                  INT$_STRW,-
                     30 0008    138                  INT$_STRL,-
                     31 0009    139                  0,-
                     00 000A    140                  INT$_STRSB,-
                     32 000B    141                  INT$_STRSW,-
                  00 33 000C    142                  0
                        000E    143
                        000E    144 DAT_STO_CMD:
                        000E    145          .BYTE   INT$_STOB,-
                     34 000E    146                  INT$_STOW,-
                     35 000F    147                  INT$_STOL,-
                     2E 0010    148                  INT$_STOL,-
                     2E 0011    149                  INT$_STSB,-
                     36 0012    150                  INT$_STSW,-
                  2E 37 0013    151                  INT$_STOL
                        0015    152
                        0015    153 DAT_TRUNC_CHK:                          ;ROUTINES TO CHECK FOR TRUNCATION
                        0015    154          .ADDRESS MAC$CK_BYT_TRU1,-     ;BYTE
                00000000' 0015  155                   MAC$CK_WRD_TRU1,-     ;WORD
```

```
                00000000'  0019   156                  0,-                          ;LONGWORD
                00000000'  001D   157                  0,-                          ;QUADWORD
                00000000'  0021   158                  MAC$CK_SBY_TRU1,-            ;SIGNED BYTE
                00000000'  0025   159                  MAC$CK_SWD_TRU1,-            ;SIGNED WORD
      00000000'00000000'   0029   160                  0                            ;OCTAWORD
                           0031   161
                           0031   162 DAT_SHIFT_FACT:                              ;SHIFT # OF ELEMENTS OF ALLOCATION
                           0031   163                                              ;BY THIS MUCH TO GET ALLOCATION
   04 01 00 03 02 01 00    0031   164          .BYTE   0,1,2,3,0,1,4                ;BYTE,WORD,LONG,QUAD,SIGNED_BYTE,
                           0038   165                                              ;SIGNED_WORD,OCTAWORD
```

```
                     0038   167 ;++
                     0038   168 ; THIS IS THE HEART OF THE MARS ASSEMBLER.  THESE ROUTINES HANDLE
                     0038   169 ; MACHINE INSTRUCTIONS WHICH APPEAR AS SPECIAL BLOCKS IN THE
                     0038   170 ; SYMBOL TABLE.  THE 'SYM$B_SEG' BYTE IS THE NUMBER OF OPERANDS
                     0038   171 ; THE INSTRUCTION NEEDS.  STARTING AT BYTE 'SYM$K_BLKSIZ' IS A
                     0038   172 ; STRING OF BYTES DESCRIBING THE OPERANDS.  THE LOW 4 BITS DEFINE
                     0038   173 ; THE SIZE OF THE OPERAND, THE NEXT 3 BITS ARE AN INDEX INTO THE
                     0038   174 ; ILLEGAL MODE TABLE, AND THE LAST BIT IS SET IF IT IS A FLOATING
                     0038   175 ; OPERAND.
                     0038   176 ;
                     0038   177 ;--
                     0038   178
                 00000000   179          .PSECT  MAC$RO_CODE_P1,NOWRT,GBL,LONG
                     0000   180
                     0000   181 STAT1::                                      ;STATEMENT = MACHINE_STAT
                     0000   182
 0000'CF    D5   0000   183          TSTL    W^MAC$GL_MOPNUM                ;WERE THERE ENOUGH OPERANDS?
       08   15   0004   184          BLEQ    10$                            ;IF LEQ YES
                0006   185          $MAC_ERR NOTENUFOPR                    ; No--set message code
    FFF2'   30   000B   186          BSBW    MAC$ERRORPT                    ;SEND ERROR MSG TO INT. FILE
            05   000E   187 10$:     RSB
```

```
                     000F   189          .SBTTL  OPCODE GENERATION
                     000F   190
                     000F   191 ;++
                     000F   192 ; FUNCTIONAL DESCRIPTION:
                     000F   193 ;
                     000F   194 ;       MINST1 IS INVOKED WHEN AN OPCODE IS ENCOUNTERED.  IT SETS
                     000F   195 ;       UP TO PROCESS THE OPERANDS THAT FOLLOW THE OPCODE.
                     000F   196 ;
                     000F   197 ; INPUTS:
                     000F   198 ;
                     000F   199 ;       MAC$GL_VALUE            SYMBOL BLOCK ADDRESS OF OPCODE
                     000F   200 ;
                     000F   201 ; OUTPUTS:
                     000F   202 ;
                     000F   203 ;       MAC$GL_MOPNUM           NUMBER OF OPERANDS FOR THIS OPCODE
                     000F   204 ;       MAC$GL_MOPPTR           POINTER TO OPERAND WORD DESCRIPTORS
                     000F   205 ;
                     000F   206 ;--
                     000F   207
                     000F   208 MINST1::                                    ;MACHINE_INST = DOPCODE
                     000F   209
  56 0000'CF   D0    000F   210          MOVL    W^MAC$GL_VALUE,R6           ;GET SYMBOL BLOCK ADDRESS
     FFE9'     30    0014   211          BSBW    MAC$CREF_OPCODE             ;CREF THE OPCODE IF NEEDED
                     0017   212          $INTOUT_WD INT$_OP,SYM$L_VAL(R6)  ;OUTPUT OPCODE TO PASS 2
                     0021   213          $INC_PC                           ;UPDATE PC FOR OPCODE
       06 A6   95    0025   214          TSTB    SYM$L_VAL+1(R6)           ;TWO-BYTE OPCODE?
          04   13    0028   215          BEQL    10$                       ;IF EQL NO
                     002A   216          $INC_PC                           ;YES--UPDATE PC FOR 2-BYTE OPCODE
       0C A6   9A    002E   217 10$:     MOVZBL  SYM$B_SEG(R6),-           ;SET UP OPERAND COUNTER
     0000'CF                0031   218                  W^MAC$GL_MOPNUM
       0D A6   9E    0034   219          MOVAB   SYM$K_BLKSIZ(R6),-        ;POINT TO OPERAND MODE WORD DESCRIPTORS
     0000'CF                0037   220                  W^MAC$GL_MOPPTR ;
                     003A   221
                     003A   222 ;
                     003A   223 ; EXIT FROM MACHINE INSTRUCTION OR OPERAND--SET FOR NEXT OPERAND
                     003A   224 ;
                     003A   225
                     003A   226 MACH_OP_EXIT:
                     003A   227                                             ; Clear Index Mode de-optimize flag,
04 AB    1010 8F    AA    003A   228          bicw2   #FLG$M_UPAFLG!FLG$M_OPTVFLIDX,4(r11) ; and DUpA flag.
     06 6B   14    E5    0040   229          BBCC    #FLG$V_CHKLPND,(R11),5$ ;CHKL PENDING?
                     0044   230          $INTOUT_X INT$_CHKL                ;YES--SEND IT NOW
        05   00    EF    004A   231 5$:      EXTZV   #OPD$V_SIZE,#OPD$S_SIZE,- ;GET SIZE OF OPERAND
  50    0000'DF          004D   232                  @W^MAC$GL_MOPPTR,R0     ;...
  0000'CF    50    D0    0051   233          MOVL    R0,W^MAC$GL_OPSIZE        ;AND STORE FOR LATER USE
        0000'CF    D4    0056   234 10$:     CLRL    W^MAC$GB_MODE             ;CLEAR MODE,IMODE,REG, AND IREG
        0000'CF    D0    005A   235          MOVL    W^MAC$GL_PSECT,-          ;START WITH CURRENT PSECT
     0000'CF                005E   236                  @W^MAC$GL_PRMSEG ;
  0000'CF    59    D1    0061   237          CMPL    R9,W^MAC$GL_INTWRNPT      ;NEAR THE END OF THE INT. BUFFER?
             03    1B    0066   238          BLEQU   20$                       ;IF LEQU NO
          FF95'    30    0068   239          BSBW    MAC$OUTFRAME              ;YES--SET UP FOR NEW BUFFER
  0000'CF    59    D0    006B   240 20$:     MOVL    R9,W^MAC$GL_EXPPTR        ;SAVE PTR TO EXPRESSION START
  0000'CF    59    D0    0070   241          MOVL    R9,W^MAC$GL_EXPEND        ;AND EXPRESSION END
   000000C4 8F    C8    0075   242          BISL2   #FLG$M_COMPEXPR!FLG$M_EXPOPT!FLG$M_EVALEXPR,-
             6B          0079   243                  (R11)                     ;ASSUME COMPILE TIME EXPRESSION,
                     007C   244                                             ; ALLOW EXPRESSION OPTIMIZATION
                     007C   245                                             ; AND EVALUATE ON PASS 2
```

```
                0000'CF  D4  007C  246  CLRL  W^MAC$GL_ABSFLAG          ;ASSUME ABSOLUTE EXPRESSION
0000'CF         0000'CF  D0  0080  247  MOVL  W^MAC$GL_PC,W^MAC$GL_SAVE_PC ;SAVE PC FOR ERROR RECOVERY
                0000'CF  D4  0087  248  CLRL  W^MAC$GL_HIGH_32          ;CLEAR HI 32 BITS IN CASE QUAD OPERAND
                         05  008B  249  RSB
```

```
                     008C   251          .SBTTL  OPERAND GENERATION
                     008C   252
                     008C   253 ;++
                     008C   254 ; FUNCTIONAL DESCRIPTION:
                     008C   255 ;
                     008C   256 ;       OPRAND IS INVOKED WHEN A REFERENCE (OPERAND) HAS BEEN SCANNED.
                     008C   257 ;       IF THERE ARE TOO MANY OPERANDS A MESSAGE IS ISSUED TO PASS 2.
                     008C   258 ;       THE MODE OF THE REFERENCE IS CHECKED TO SEE IF IT IS LEGAL FOR
                     008C   259 ;       THIS OPERAND.  THE REFERENCE IS THEN EMITTED TO PASS 2.
                     008C   260 ;
                     008C   261 ; INPUTS:
                     008C   262 ;
                     008C   263 ;       MAC$GL_MOPPTR           POINTER TO OPERAND WORD DESCRIPTOR
                     008C   264 ;       MAC$GB_MODE             MODE OF OPERAND
                     008C   265 ;
                     008C   266 ; OUTPUTS:
                     008C   267 ;
                     008C   268 ;       THE INTERMEDIATE CODE FOR THIS OPERAND IS EMITTED TO THE
                     008C   269 ;       INTERMEDIATE FILE.
                     008C   270 ;
                     008C   271 ;--
                     008C   272
                     008C   273 OPRAND::                                     ;OPERANDS = REF
                     008C   274                                             ;OPERANDS = OPERANDS DCOMMA REF
                     008C   275
          0000'CF    D5  008C   276          TSTL    W^MAC$GL_MOPNUM         ;SHOULD WE REALLY BE HERE?
                11   14  0090   277          BGTR    10$                     ;IF GTR THEN CONTINUE
                          0092   278          $MAC_ERR TOOMNYOPND             ; Else set error message code
             FF66'   30  0097   279          BSBW    MAC$ERRORPX             ;SEND ERROR TO PASS 2
0000'CF      0000'CF   D0  009A   280          MOVL    W^MAC$GL_SAVE_PC,W^MAC$GL_PC ;RESET PC TO NOT COUNT OPERAND
                97   11  00A1   281          BRB     MACH_OP_EXIT            ;FINISH UP THIS OPERAND
          56    0000'DF   3C  00A3   282 10$:    MOVZWL  @W^MAC$GL_MOPPTR,R6     ;GET OPERAND DESC. WORD THIS OPRAND
                05    05   EF  00A8   283          EXTZV   #OPD$V_MODE,#OPD$S_MODE,- ;GET THE OPERAND MODE
                55    56        00AB   284                  R6,R5               ;INTO R5
          54    0000'CF   9A  00AD   285          MOVZBL  W^MAC$GB_MODE,R4        ;GET OPERAND MODE WE SCANNED
50    00000000'EF45   3C  00B2   286          MOVZWL  L^MAC$AW_ILLMODTB[R5],R0 ;GET TABLE ENTRY FOR ACCESS MODE
            14 50     54   E1  00BA   287          BBC     R4,R0,20$               ;BRANCH IF LEGAL MODE
                          00BE   288          $MAC_ERR ILLMODE                ; No--get message code
                05    54   91  00C3   289          CMPB    R4,#ADM$_REGISTER       ; Is addressing mode register?
                      05   12  00C6   290          BNEQ    14$                     ; No if NEQ
                   FF35'   30  00C8   291          BSBW    MAC$ERRORPX             ;SEND ERROR TO PASS 2
                      03   11  00CB   292          BRB     16$
                          00CD   293 14$:
                   FF30'   30  00CD   294          BSBW    MAC$ERRORPT             ; Send error to pass-2
                          00D0   295 16$:
                      56   D4  00D0   296          CLRL    R6                      ;USE ZERO DESCRIPTOR
             0000'CF  02   C0  00D2   297 20$:    ADDL2   #2,W^MAC$GL_MOPPTR      ;ADVANCE TO NEXT DESCRIPTOR
                0000'CF   D7  00D7   298          DECL    W^MAC$GL_MOPNUM         ;DECREMENT OPERAND COUNT
                      0F   12  00DB   299          BNEQ    30$                     ;IF NEQ THEN NOT LAST OPERAND
                0000'CF   B1  00DD   300          CMPW    W^MAC$GL_ERRPTX,-       ;LAST OPERAND--FIRST ON LINE?
                0000'8F         00E1   301                  #MAC$AB_LINEBF          ;
                      06   13  00E4   302          BEQL    30$                     ;IF EQL YES
                      0D   E3  00E6   303          BBCS    #OPF$V_LASTOPR,-        ;NO--MARK LAST OPERAND
                00 0000'CF        00E8   304                  W^MAC$GL_OPSIZE,30$
                04 6B     02   E0  00EC   305 30$:    BBS     #FLG$V_COMPEXPR,(R11),40$ ;BRANCH IF OPTIMIZABLE
                0D 6B     07   E5  00F0   306          BBCC    #FLG$V_EXPOPT,(R11),50$ ;ELSE FLAG UNABLE TO OPTIMIZE
                09 6B     07   E1  00F4   307 40$:    BBC     #FLG$V_EXPOPT,(R11),50$ ;BRANCH IF UNABLE TO OPTIMIZE
```

```
                FF05'  30   00F8   308          BSBW    MAC$OPTIMIZEXPR          ;OPTIMIZE EXPRESSION
                  0C   E3   00FB   309          BBCS    #OPF$V_OPTEXP,-          ;MARK OPTIMIZED
      00 0000'CF       00FD   310                          W^MAC$GL_OPSIZE,50$
00A1 8F    56   B1   0101   311  50$:    CMPW    R6,#OPD$M_BB             ;BRANCH DESTINATION?
                  0E   13   0106   312          BEQL    60$              ;IF EQL YES
00C2 8F    56   B1   0108   313          CMPW    R6,#OPD$M_BW             ;BRANCH DESTINATION?
                  03   13   010D   314          BEQL    55$              ;IF EQL YES
                008E   31   010F   315          BRW     120$             ;ELSE NOT A BRANCH DESTINATION
                       0112   316  55$:    $INC_PC                          ;YES--UPDATE PC FOR BRANCH WORD
0F    0000'CF    91   0116   317  60$:    CMPB    W^MAC$GB_VAL3,#REG$_PC   ;REGISTER MUST BE 'PC'
                  0B   13   011B   318          BEQL    70$              ;IF EQL OK
                       011D   319          $MAC_ERR ILLBRDEST               ; Illegal branch destination
                FEDB'  30   0122   320          BSBW    MAC$ERRORPX              ;SEND ERROR TO PASS 2
                00F4   31   0125   321          BRW     150$             ;FINISH
0A    0000'CF    91   0128   322  70$:    CMPB    W^MAC$GB_MODE,#ADM$_BYTE_DISP ;CORRECT BRANCH SIZE
                  06   12   012D   323          BNEQ    80$
                       012F   324          $DEC_PC                          ;
                  13   11   0133   325          BRB     100$             ;JOIN COMMON CODE
0C    0000'CF    91   0135   326  80$:    CMPB    W^MAC$GB_MODE,#ADM$_WORD_DISP
                  07   12   013A   327          BNEQ    90$
                       013C   328          $DEC_PC #2
                  05   11   0141   329          BRB     100$
                       0143   330  90$:    $DEC_PC #4
                  7E   94   0148   331  100$:   CLRB    -(SP)                    ;ASSUME NOT OPTIMIZED
53    0000'CF    D0   014A   332          MOVL    W^MAC$GL_EXPOPVL1,R3     ;GET (MAYBE) OPTIMIZED VALUE
  37 6B    07   E0   014F   333          BBS     #FLG$V_EXPOPT,(R11),110$ ;BRANCH IF WE OPTIMIZED
                  53   D4   0153   334          CLRL    R3               ;ASSUME GLOBAL
52    0000'CF    D0   0155   335          MOVL    W^MAC$GL_EXPPTR,R2       ;GET EXPRESSION POINTER
50    0000'CF    52   C3   015A   336          SUBL3   R2,W^MAC$GL_EXPEND,R0    ;COMPUTE SIZE OF EXPRESSION
                       0160   337  104$:
                  28   13   0160   338          BEQL    110$             ;IF EQL NO EXPRESSION
                06    50   D1   0162   339          CMPL    R0,#6            ;6 BYTES?
                  11   13   0165   340          BEQL    106$             ; Yes if EQL
             17   01 A2   91   0167   341          CMPB    1(R2),#INT$_NEWL         ; Is it a new-line?
                  1D   12   016B   342          BNEQ    110$             ; No if NEQ
                51    62   9A   016D   343          MOVZBL  (R2),R1          ; Get frame length
                52    51   C0   0170   344          ADDL2   R1,R2            ; Point to next frame
                50    51   C2   0173   345          SUBL2   R1,R0            ; and reduce size of expression
                  E8   11   0176   346          BRB     104$
                       0178   347  106$:
             2D   01 A2   91   0178   348          CMPB    1(R2),#INT$_STKS         ;YES--STACK SYMBOL REFERENCE?
                  0C   12   017C   349          BNEQ    110$             ;IF NEQ NO
             53   02 A2   D0   017E   350          MOVL    2(R2),R3         ;YES--GET ID ADDRESS
6E    0000'CF    90   0182   351          MOVB    W^MAC$GL_PSECT,(SP)      ;MUST BE IN SAME PSECT
                02 A2   D4   0187   352          CLRL    2(R2)            ;FLAG SPECIAL RESOLUTION
                50    09   9A   018A   353  110$:   MOVZBL  #9,R0             ;WE WILL OUTPUT 9 BYTES
                  FE70'  30   018D   354          BSBW    MAC$INTOUT_N             ;MAKE ROOM FOR THEM
                89    0E   90   0190   355          MOVB    #INT$_BDST,(R9)+         ;STORE INT. CODE
89    0000'CF    B0   0193   356          MOVW    W^MAC$GL_OPSIZE,(R9)+    ;STORE FLAGS
                89    53   D0   0198   357          MOVL    R3,(R9)+         ;STORE 0 OR SYMBOL ID ADDRESS
                89    8E   90   019B   358          MOVB    (SP)+,(R9)+              ;STORE 0 OR PSECT NUMBER
                  7C   11   019E   359          BRB     150$
                       01A0   360  ;
                       01A0   361  ; NOT BRANCH DESTINATION
                       01A0   362  ;
  14 6B    24   E1   01A0   363  120$:   BBC     #FLG$V_UPAFLG,(R11),125$ ;BRANCH IF DUPA WAS NOT SEEN
0F    0000'CF    91   01A4   364          CMPB    W^MAC$GB_REG,#REG$_PC    ;YES--IS REGISTER PC?
```

```
                 0D   12  01A9   365          BNEQ    125$                            ;IF NEQ NO
            0A   54   91  01AB   366          CMPB    R4,#ADM$_BYTE_DISP              ;YES--IS MODE LEGAL?
                 08   19  01AE   367          BLSS    125$                            ;IF LSS YES
                      01B0   368          $MAC_ERR OPRNDSYNX                      ;NO--TELL OF OPERAND SYNTAX ERROR
               FE48'  30  01B5   369          BSBW    MAC$ERRORPT                     ;...
        1A 6B    07   E1  01B8   370 125$:    BBC     #FLG$V_EXPOPT,(R11),130$ ;BRANCH IF CANNOT OPTIMIZE
            50   0C   9A  01BC   371          MOVZBL  #12,R0                          ;SET TO STORE 12 BYTES
               FE3E'  30  01BF   372          BSBW    MAC$INTOUT_N                    ;SET UP FOR IT
            89   1E   90  01C2   373          MOVB    #INT$_REF,(R9)+                 ;STORE INT. CODE
     89     0000'CF   D0  01C5   374          MOVL    W^MAC$GL_VALUE,(R9)+            ;STORE REGISTERS/MODES
     89     0000'CF   B0  01CA   375          MOVW    W^MAC$GL_OPSIZE,(R9)+           ;STORE FLAGS
     89     0000'CF   D0  01CF   376          MOVL    W^MAC$GL_EXPOPVL1,(R9)+ ;STORE OPTIMIZED VALUE
                 13   11  01D4   377          BRB     140$
            50   08   9A  01D6   378 130$:    MOVZBL  #8,R0                           ;SET TO STORE 8 BYTES
               FE24'  30  01D9   379          BSBW    MAC$INTOUT_N                    ;SET UP FOR IT
            89   1E   90  01DC   380          MOVB    #INT$_REF,(R9)+                 ;STORE INT. CODE
     89     0000'CF   D0  01DF   381          MOVL    W^MAC$GL_VALUE,(R9)+            ;STORE MODES/REGISTERS
     89     0000'CF   B0  01E4   382          MOVW    W^MAC$GL_OPSIZE,(R9)+           ;STORE FLAGS
                      01E9   383 140$:
     01     0000'CF   91  01E9   384          CMPB    W^MAC$GL_VALUE,#ADM$_IMMEDIATE ; Is address mode immediate?
                 2C   12  01EE   385          BNEQ    150$                            ; No if NEQ
     08     0000'CF   91  01F0   386          CMPB    W^MAC$GL_OPSIZE,#8              ; Is operand a QUAD or OCTA value?
                 25   19  01F5   387          BLSS    150$                            ; No if LSS
                      01F7   388          $INTOUT_LW INT$_STIL,<W^MAC$GL_HIGH_32> ; Output bits 32-63
     10     0000'CF   91  0201   389          CMPB    W^MAC$GL_OPSIZE,#16             ; Is operand an OCTA value?
                 14   12  0206   390          BNEQ    150$                            ; No if NEQ
                      0208   391          $INTOUT_LW INT$_STIL,<W^MAC$GQ_HIGH_64+0> ; Output bits 64-95
                      0212   392          $INTOUT_LW INT$_STIL,<W^MAC$GQ_HIGH_64+4> ; and then bits 96-127
0000'CF    02   90  021C   393 150$:    MOVB    #RDX$V_DECIMAL,W^MAC$GB_RDXNDX ;RESET RADIX
          FE16   31  0221   394          BRW     MACH_OP_EXIT
```

```
                     0224   396          .SBTTL  ASSIGNMENT STATMENTS
                     0224   397
                     0224   398 ;++
                     0224   399 ; FUNCTIONAL DESCRIPTION:
                     0224   400 ;
                     0224   401 ;       THESE ROUTINES ARE INOVKED WHEN AN ASSIGNMENT STATMENT
                     0224   402 ;       IS DETECTED.  IF ENTRY AT ASSHD3, IT IS FLAGGED AS AN
                     0224   403 ;       ASSIGNMENT TO 'PC'.  IF ENTRY AT ASSHD2, THE SYMBOL
                     0224   404 ;       IS FLAGGED AS GLOBAL.
                     0224   405 ;
                     0224   406 ; INPUTS:
                     0224   407 ;
                     0224   408 ;       MAC$AL_VALSTACK-8[R7]   (ASSHD2) SYMBOL BLOCK OF ID
                     0224   409 ;       MAC$AL_VALSTACK-4[R7]   (ASSHD1) SYMBOL BLOCK OF ID
                     0224   410 ;
                     0224   411 ; OUTPUTS:
                     0224   412 ;
                     0224   413 ;       MAC$GL_ASNPTR           POINTER TO SYMBOL BLOCK OF ID
                     0224   414 ;       MAC$GL_OPSIZE           4
                     0224   415 ;
                     0224   416 ;--
                     0224   417
                     0224   418
                     0224   419 ASSHD3::                                    ;ASSIGN_HEAD = DPC
           50   FF 8F   98  0224   420          CVTBL   #-1,R0              ;MARK PC AUGMENTATION
                  18   11  0228   421          BRB     ASSIGN_HEAD
                     022A   422
                     022A   423 ASSHD2::                                    ;ASSIGN_HEAD = ID DEQ DEQ
        50   FFF8'CF47   D0  022A   424          MOVL    W^MAC$AL_VALSTACK-8[R7],R0 ;POINT TO ID SYMBOL BLOCK
           09 A0    04   A8  0230   425          BISW2   #SYM$M_GLOBL,SYM$W_FLAG(R0) ;MARK SYMBOL AS GLOBAL
        09 A0    0800 8F   A8  0234   426          BISW2   #SYM$M_RELPSECT,SYM$W_FLAG(R0)  ;ALWAYS OUTPUT GLOBAL SYMBOL
                  06   11  023A   427          BRB     ASSIGN_HEAD
                     023C   428
                     023C   429 ASSHD1::                                    ;ASSIGN_HEAD = ID DEQ
     50   FFFC'CF47   D0  023C   430          MOVL    W^MAC$AL_VALSTACK-4[R7],R0 ;POINT TO ID SYMBOL BLOCK
                     0242   431 ASSIGN_HEAD:
      0000'CF   50   D0  0242   432          MOVL    R0,W^MAC$GL_ASNPTR          ;SAVE POINTER TO ID
          0000'CF    D4  0247   433          CLRL    W^MAC$GL_PRVSEG             ;ALLOW EXPRESSION IN ANY SEGMENT
        00 6B    06   E5  024B   434          BBCC    #FLG$V_EVALEXPR,(R11),10$ ;DON'T EVALUATE EXPRESSION
       00002004 8F   C8  024F   435 10$:     BISL2   #FLG$M_COMPEXPR!FLG$M_OPRND,- ;ASSUME COMPILE TIME EXPR
                  6B       0255   436                      (R11)               ;AND FLAG IN OPERAND FIELD
          0000'CF    D4  0256   437          CLRL    W^MAC$GL_ABSFLAG            ;ASSUME ABSOLUTE EXPRESSION
      0000'CF   04   9A  025A   438          MOVZBL  #4,W^MAC$GL_OPSIZE          ;SET OPERAND MAX SIZE TO 4 BYTES
                     025F   439 ;
                     025F   440 ; IF CREFFING, SAVE LINE/PAGE SO THEY ARE CORRECT
                     025F   441 ;
        21 6B    1E   E1  025F   442          BBC     #FLG$V_CRF,(R11),30$    ;BRANCH IF NOT CREFFING
          0000'CF    D0  0263   443          MOVL    W^MAC$GL_SRCPAG,-       ;YES--SAVE SOURCE PAGE
          0000'CF        0267   444                  W^MAC$GL_SAV_PAG
          0000'CF    D0  026A   445          MOVL    W^MAC$GL_LINBAS,-       ;SAVE LINE BASE
          0000'CF        026E   446                  W^MAC$GL_SAV_BAS
     50   0000'CF    D0  0271   447          MOVL    W^MAC$GL_LINENUM,R0     ;GET THE LINE NUMBER
        05 6B    19   E1  0276   448          BBC     #FLG$V_SEQFIL,(R11),20$ ;BRANCH IF NOT SEQUENCED
     50   0000'CF    D0  027A   449          MOVL    W^MAC$GL_RECHDBUF,R0    ;YES--GET SEQUENCE NUMBER
     0000'CF   50   D0  027F   450 20$:     MOVL    R0,W^MAC$GL_SAV_LIN     ;AND SAVE LINE NUMBER
                     05  0284   451 30$:     RSB
```

```
                     0285   453 ;++
                     0285   454 ; FUNCTIONAL DESCRIPTION:
                     0285   455 ;
                     0285   456 ;       ASSGN1 IS INVOKED TO FINISH PROCESSING AN ASSIGNMENT STATEMENT.
                     0285   457 ;       THE EXPRESSION HAS BEEN EVALUATED, AND IS ON THE VALUE STACK.
                     0285   458 ;       IF THE ASSIGNMENT IS TO THE PC, CODE IS EMITTED TO THE INTERMEDIATE
                     0285   459 ;       FILE TO AUGMENT THE PC.  IF THE ASSIGNMENT IS NOT TO PC, A
                     0285   460 ;       CHECK IS MADE FOR A MULTIPLE LABEL DEFINITION, AND THEN THE
                     0285   461 ;       FLAGS IN THE SYMBOL BLOCK ARE UPDATED.  CODE IS EMITTED TO
                     0285   462 ;       THE INTERMEDIATE FILE TO UPDATE THE SYMBOL BLOCK IN PASS 2.
                     0285   463 ;
                     0285   464 ; INPUTS:
                     0285   465 ;
                     0285   466 ;       MAC$GL_ASNPTR           (-1) IF PC AUGMENTATION, ELSE POINTER
                     0285   467 ;                               TO SYMBOL BLOCK OF ID.
                     0285   468 ;       MAC$AL_VALSTACK-4[R7]   EXPRESSION VALUE
                     0285   469 ;
                     0285   470 ; OUTPUTS:
                     0285   471 ;
                     0285   472 ;
                     0285   473 ;--
                     0285   474
                     0285   475 ASSGN1::                                    ;ASSIGNMENT = ASSIGN_HEAD EXPR DEOL
 56   0000'CF   D0   0285   476         MOVL    W^MAC$GL_ASNPTR,R6          ;GET POINTER TO ID SYMBOL BLOCK
   08 6B   02   E0   028A   477         BBS     #FLG$V_COMPEXPR,(R11),10$ ;MUST BE COMPILE TIME EXPRESSION
                028E   478         $MAC_ERR ASGNMNTSYN                     ; No--send message to pass 2
        FD6A'   30   0293   479         BSBW    MAC$ERRORPT
50   56   01   C1   0296   480 10$:    ADDL3   #1,R6,R0                    ;IS THIS PC ASSIGNMENT (R6=-1?)?
          22   12   029A   481         BNEQ    20$                         ;IF NEQ NO
        FD61'   30   029C   482         BSBW    MAC$SET_PC                  ;YES--RECORD HI MARK OF PC
   56   FFFC'CF47   D0   029F   483         MOVL    W^MAC$AL_VALSTACK-4[R7],R6 ;GET NEW VALUE
55   56   0000'CF   C3   02A5   484         SUBL3   W^MAC$GL_PC,R6,R5           ;COMPUTE AUGMENTATION
                02AB   485         $INTOUT_LW INT$_AUGPC,R5                ;SEND TO PASS 2
     0000'CF   56   D0   02B3   486         MOVL    R6,W^MAC$GL_PC              ;SET NEW PC
        FD45'   30   02B8   487         BSBW    MAC$SET_PC                  ;CHECK NEW PC
          008D   31   02BB   488         BRW     80$                         ;
                02BE   489 ;
                02BE   490 ; EXPRESSION DOES NOT INVOLVE PC
                02BE   491 ;
   08 09 A6   03   E1   02BE   492 20$:    BBC     #SYM$V_EXTRN,SYM$W_FLAG(R6),30$ ;EXTERNAL?
                02C3   493         $MAC_ERR SYMDCLEXTR                     ; Yes-error
        FD35'   30   02C8   494         BSBW    MAC$ERRORPT                 ;ISSUE ERROR TO PASS 2
          00AE   30   02CB   495 30$:    BSBW    MAC$MUL_DEF_CHK             ;SEE IF MULTIPLY DEFINED
 0C A6   0000'CF   90   02CE   496         MOVB    W^MAC$GL_PRVSEG,SYM$B_SEG(R6) ;DEFINE IN EXPRESSION PSECT
     09 A6   10   AA   02D4   497         BICW2   #SYM$M_ABS,SYM$W_FLAG(R6) ;ASSUME NOT ABSOLUTE
     0000'CF   D5   02D8   498         TSTL    W^MAC$GL_ABSFLAG            ;IS EXPRESSION ABSOLUTE?
          08   12   02DC   499         BNEQ    50$                         ;IF NEQ NO
        0C A6   94   02DE   500         CLRB    SYM$B_SEG(R6)               ;YES--MAKE ABSOLUTE PSECT
   00 09 A6   04   E3   02E1   501         BBCS    #SYM$V_ABS,SYM$W_FLAG(R6),50$ ;SET ABSOLUTE FLAG
   09 09 A6   06   E0   02E6   502 50$:    BBS     #SYM$V_LOCAL,SYM$W_FLAG(R6),60$ ;IS SYMBOL LOCAL?
     04 0005'CF   E9   02EB   503         BLBC    W^ENB$G_DEBUG+SYM$L_VAL,60$     ;NO--BRANCH IF NO ENABLE DEBUG
     09 A6   20   A8   02F0   504         BISW2   #SYM$M_DEBUG,SYM$W_FLAG(R6) ;LET DEBUGGER KNOW ABOUT SYMBOL
05 A6   FFFC'CF47   D0   02F4   505 60$:    MOVL    W^MAC$AL_VALSTACK-4[R7],- ;PUT IN SYMBOL VALUE
                02FB   506                             SYM$L_VAL(R6)
        0101 8F   A8   02FB   507         BISW2   #SYM$M_DEF!SYM$M_ASN,-  ;MARK AS DEFINED BY ASSIGNMENT
          09 A6   02FF   508                             SYM$W_FLAG(R6)
     55   00'8F   9A   0301   509         MOVZBL  #CRF$K_DEF,R5               ;SET DEFINITION FLAG
```

```
                0000'CF    DD  0305   510          PUSHL   W^MAC$GL_LINBAS          ;GET READY TO SET RIGHT LINE/PAGE
                0000'CF    DD  0309   511          PUSHL   W^MAC$GL_LINENUM         ;BY SAVING CURRENT VALUES
                0000'CF    DD  030D   512          PUSHL   W^MAC$GL_SRCPAG          ;...
                0000'CF    DD  0311   513          PUSHL   W^MAC$GL_RECHDBUF
                0000'CF    D0  0315   514          MOVL    W^MAC$GL_SAV_BAS,-       ;NOW SET VALUES WE WANT
                0000'CF        0319   515                  W^MAC$GL_LINBAS
          50    0000'CF    D0  031C   516          MOVL    W^MAC$GL_SAV_LIN,R0
    0000'CF     50    D0  0321   517          MOVL    R0,W^MAC$GL_LINENUM
    0000'CF     50    D0  0326   518          MOVL    R0,W^MAC$GL_RECHDBUF     ;(IN CASE SEQUENCED FILE)
                0000'CF    D0  032B   519          MOVL    W^MAC$GL_SAV_PAG,-
                0000'CF        032F   520                  W^MAC$GL_SRCPAG
                   FCCB'    30  0332   521          BSBW    MAC$CREF_SYM             ;OUTPUT TO CREF IF CREFFING
                0000'CF  8ED0  0335   522          POPL    W^MAC$GL_RECHDBUF        ;RESTORE OLD LINES/PAGES
                0000'CF  8ED0  033A   523          POPL    W^MAC$GL_SRCPAG          ;
                0000'CF  8ED0  033F   524          POPL    W^MAC$GL_LINENUM         ;
                0000'CF  8ED0  0344   525          POPL    W^MAC$GL_LINBAS          ;
                      10    10  0349   526          BSBB    MAC$INTOUT_ASN           ;OUTPUT ASN TO INTERMED. FILE
          00 6B     06    E3  034B   527  80$:    BBCS    #FLG$V_EVAL_EXPR,(R11),90$ ;ALLOW EXPRESSION EVALUATION
                              034F   528  90$:    $INTOUT_LW INT$_PRIL,<W^MAC$AL_VALSTACK-4[R7]> ;PRINT EXPRESSION
                           05  035A   529          RSB
```

```
                        035B   531 ;++
                        035B   532 ; FUNCTIONAL DESCRIPTION:
                        035B   533 ;
                        035B   534 ;       THIS ROUTINE OUTPUTS AN ASSIGN COMMAND AND DATA TO THE
                        035B   535 ;       INTERMEDIATE FILE.
                        035B   536 ;
                        035B   537 ; INPUTS:
                        035B   538 ;
                        035B   539 ;       R6      POINTS TO SYMBOL BLOCK
                        035B   540 ;
                        035B   541 ;--
                        035B   542
                        035B   543 MAC$INTOUT_ASN::
        50    0C    9A  035B   544           MOVZBL  #12,R0                  ;SIZE OF AN ASN COMMAND AND DATA
           FC9F'    30  035E   545           BSBW    MAC$INTOUT_N            ;MAKE ROOM FOR IT
        89    0C    90  0361   546           MOVB    #INT$_ASN,(R9)+         ;STORE THE COMMAND
        89    56    D0  0364   547           MOVL    R6,(R9)+                ;STORE SYMBOL BLOCK ADDRESS
     89    0C A6    90  0367   548           MOVB    SYM$B_SEG(R6),(R9)+     ;STORE SYMBOL SEGMENT
     89    05 A6    D0  036B   549           MOVL    SYM$L_VAL(R6),(R9)+     ;STORE SYMBOL VALUE
           50       D4  036F   550           CLRL    R0                      ;ASSUME ABSOLUTE
  02 09 A6    04    E0  0371   551           BBS     #SYM$V_ABS,SYM$W_FLAG(R6),10$ ;BRANCH IF ABSOLUTE
           50       D6  0376   552           INCL    R0                      ;NO--MAKE RELOCATABLE
        89    50    90  0378   553 10$:      MOVB    R0,(R9)+                ;STORE ABS/REL FLAG
              05       037B   554           RSB
                        037C   555
                        037C   556 ;++
                        037C   557 ; FUNCTIONAL DESCRIPTION:
                        037C   558 ;
                        037C   559 ;       THIS ROUTINE CHECKS FOR A MULTIPLY DEFINED LABEL.  IF THE
                        037C   560 ;       LABEL IS MULTIPLY DEFINED, AN ERROR MESSAGE IS ISSUED TO
                        037C   561 ;       PASS 2.
                        037C   562 ;
                        037C   563 ; INPUTS:
                        037C   564 ;
                        037C   565 ;       R6      SYMBOL BLOCK ADDRESS
                        037C   566 ;
                        037C   567 ;--
                        037C   568
                        037C   569 MAC$MUL_DEF_CHK::
  0D 09 A6    00    E1  037C   570           BBC     #SYM$V_DEF,SYM$W_FLAG(R6),10$ ;BRANCH IF NOT DEFINED
  08 09 A6    08    E0  0381   571           BBS     #SYM$V_ASN,SYM$W_FLAG(R6),10$ ;BRANCH IF BY ASSIGNMENT
                        0386   572           $MAC_ERR MULDEFLBL              ; This is multiply defined
           FC72'    31  038B   573           BRW     MAC$ERRORPT             ;ISSUE ERROR TO PASS 2
     06 0005'CF    E9  038E   574 10$:      BLBC    W^ENB$G_SUPPRESS+SYM$L_VAL,20$ ;BRANCH IF NOT ENABLE SUPPRESSION
09 A6    4000 8F    A8  0393   575           BISW2   #SYM$M_SUPR,SYM$W_FLAG(R6) ;YES--SET SUPPRESS BIT THIS SYMBOL
              05       0399   576 20$:      RSB
```

```
                                                     H 8
                     039A   578          .SBTTL  BLOCK DATA STORAGE DIRECTIVES
                     039A   579
                     039A   580 ;++
                     039A   581 ; FUNCTIONAL DESCRIPTION:
                     039A   582 ;
                     039A   583 ;       THESE ROUTINES (BLKBYT, BLKWRD, BLKLNG, BLKQUD AND BLKOCT) ARE
                     039A   584 ;       CALLED WHEN A BLOCK DATA DIRECTIVE IS SCANNED.  THE
                     039A   585 ;       SHIFT COUNT FOR THE PARTICULAR DATA TYPE IS SET INTO
                     039A   586 ;       MAC$GL_VALUE AND FLAGS ARE SET TO SCAN THE EXPRESSION
                     039A   587 ;       INDICATED THE NUMBER OF UNITS OF STORAGE TO ALLOCATE.
                     039A   588 ;
                     039A   589 ; OUTPUTS:
                     039A   590 ;
                     039A   591 ;       MAC$GL_VALUE    SHIFT COUNT TO SHIFT NUMBER OF UNITS INTO BYTES
                     039A   592 ;       MAC$GL_FLAGS    FLG$M_COMPEXPR IS SET (LOOK FOR COMPILE TIME EXPRESSION
                     039A   593 ;                       FLG$M_EVALEXPR IS CLEARED (DON'T EVALUATE ON PASS 2)
                     039A   594 ;
                     039A   595 ;--
                     039A   596
                     039A   597          .ENABL  LSB
                     039A   598
                     039A   599 BLKBYT::                                 ;BLOCK_TYPE = KBLKB
  0D   10            039A   600          BSBB    10$                     ;GO TO COMMON ROUTINE
  00                 039C   601          .BYTE   0                       ;SHIFT ALLOCATION 0
                     039D   602
                     039D   603 BLKWRD::                                 ;BLOCK_TYPE = KBLKW
  0A   10            039D   604          BSBB    10$                     ;GO TO COMMON ROUTINE
  01                 039F   605          .BYTE   1                       ;SHIFT ALLOCATION ONCE
                     03A0   606
                     03A0   607 BLKLNG::                                 ;BLOCK_TYPE = KBLKL
                     03A0   608                                          ;       OR    KLBKA
                     03A0   609                                          ;       OR    KBLKF
  07   10            03A0   610          BSBB    10$                     ;GO TO COMMON ROUTINE
  02                 03A2   611          .BYTE   2                       ;SHIFT ALLOCATION TWICE
                     03A3   612
                     03A3   613 BLKQUD::                                 ;BLOCK_TYPE = KBLKQ
                     03A3   614                                          ;       OR    KBLKD
                     03A3   615                                          ;       OR      KBLKG
  04   10            03A3   616          BSBB    10$                     ;GO TO COMMON ROUTINE
  03                 03A5   617          .BYTE   3                       ;SHIFT ALLOCATION THREE TIMES
                     03A6   618
                     03A6   619 BLKOCT::                                 ;BLOCK TYPE = KBLKO
                     03A6   620                                          ;       OR    KBLKH
  01   10            03A6   621          BSBB    10$                     ;GOTO COMMON ROUTINE
  04                 03A8   622          .BYTE   4                       ;SHIFT ALLOCATION FOUR TIMES
                     03A9   623
0000'CF   9E   9A    03A9   624 10$:     MOVZBL  @(SP)+,W^MAC$GL_VALUE   ;SET SHIFT COUNT AS VALUE
     6B   04   C8    03AE   625          BISL2   #FLG$M_COMPEXPR,(R11)   ;LOOK FOR COMPILE TIME EXPRESSION
  00 6B   06   E5    03B1   626          BBCC    #FLG$V_EVALEXPR,(R11),.+1 ;DON'T OUTPUT EXPRESSION TO PASS 2
     0000'CF   D4    03B5   627          CLRL    W^MAC$GL_ABSFLAG        ;LOOK FOR ABSOLUTE EXPRESSION
               05    03B9   628          RSB
                     03BA   629
                     03BA   630          .DSABL  LSB
```

```
                          03BA    632 ;++
                          03BA    633 ; FUNCTIONAL DESCRIPTION:
                          03BA    634 ;
                          03BA    635 ;       THESE TWO ROUTINES (BSTAT1 AND BSTAT2) FINISH PROCESSING OF
                          03BA    636 ;       BLOCK DATA DIRECTIVES.
                          03BA    637 ;
                          03BA    638 ;--
                          03BA    639
                          03BA    640         .ENABL  LSB
                          03BA    641
                          03BA    642 BSTAT1::                                    ;BLOCK_STAT = BLOCK_TYPE
          56    01    9A  03BA    643         MOVZBL  #1,R6                       ;DEFAULT ALLOCATION IS 1 UNIT
    55  0000'CF47   D0  03BD    644         MOVL    W^MAC$AL_VALSTACK[R7],R5;GET THE SHIFT COUNT
              0C    11  03C3    645         BRB     10$
                          03C5    646
                          03C5    647 BSTAT2::                                    ;BLOCK_STAT = BLOCK_TYPE EXPR
    56  0000'CF47   D0  03C5    648         MOVL    W^MAC$AL_VALSTACK[R7],R6;GET NUMBER OF UNITS OF ALLOCATION
    55  FFFC'CF47   D0  03CB    649         MOVL    W^MAC$AL_VALSTACK-4[R7],R5 ;GET THE SHIFT COUNT
            FC2C'   30  03D1    650 10$:    BSBW    MAC$SET_PC                  ;RECORD HIGH PC
          0000'CF   D5  03D4    651         TSTL    W^MAC$GL_ABSFLAG            ;EXPRESSION MUST BE ABSOLUTE
              0A    13  03D8    652         BEQL    20$                         ;IF EQL IT IS
                          03DA    653         $MAC_ERR BLKEXPNABS                 ; Set message code
            FC1E'   30  03DF    654         BSBW    MAC$ERRORPT                 ;ISSUE MESSAGE TO PASS 2
              55    7C  03E2    655         CLRQ    R5                          ;ALLOCATE NO SPACE
  56    56    55    78  03E4    656 20$:    ASHL    R5,R6,R6                    ;CONVERT ALLOCATION TO BYTES
                          03E8    657         $INTOUT_LW INT$_AUGPC,R6            ;AUGMENT PC BY THAT MANY BYTES
                          03F0    658         $INC_PC R6                          ;INCREMENT PC FOR PASS 1 ALSO
                          03F5    659         $INTOUT_LW INT$_PRIL,<W^MAC$GL_PC> ;LIST NEW PC
    00 6B   06    E3  03FF    660         BBCS    #FLG$V_EVALEXPR,(R11),30$ ;ALLOW EXPRESSION EVALUATION ON PASS 2 AGA
              05    0403    661 30$:    RSB
                          0404    662
                          0404    663         .DSABL  LSB
```

```
                     0404   665          .SBTTL  LABEL DEFINITIONS
                     0404   666
                     0404   667          .ENABL  LSB
                     0404   668
                     0404   669 ;++
                     0404   670 ; FUNCTIONAL DESCRIPTION:
                     0404   671 ;
                     0404   672 ;       THESE ROUTINES DEFINE LABELS.  IF ENTRY IS AT LBL2 THE LABEL
                     0404   673 ;       IS DEFINED GLOBALLY. IF ENTRY IS AT LBL1 THE LABEL IS DEFINED
                     0404   674 ;       AS A LOCAL LABEL.
                     0404   675 ;
                     0404   676 ; INPUTS:
                     0404   677 ;
                     0404   678 ;       MAC$AL_VALSTACK-8[R7]   (LBL2) SYMBOL BLOCK ADDRESS OF ID
                     0404   679 ;       MAC$AL_VALSTACK-4[R7]   (LBL1) SYMBOL BLOCK ADDRESS OF ID
                     0404   680 ;
                     0404   681 ; OUTPUTS:
                     0404   682 ;
                     0404   683 ;       MAC$GL_LSB              INCREMENTED IF NOT LOCAL LABEL
                     0404   684 ;                               AND 'ENABL LSB'
                     0404   685 ;
                     0404   686 ;--
                     0404   687
                     0404   688          .ENABL  LSB
                     0404   689
                     0404   690 LBL2::                                       ;LABEL = ID DCOLON DCOLON
 56   FFF8'CF47   D0  0404   691          MOVL    W^MAC$AL_VALSTACK-8[R7],R6 ;POINT TO ID SYMBOL BLOCK
    09 A6    04   A8  040A   692          BISW2   #SYM$M_GLOBL,SYM$W_FLAG(R6) ;MARK AS GLOBAL SYMBOL
09 A6    0800 8F   A8  040E   693          BISW2   #SYM$M_RELPSECT,SYM$W_FLAG(R6)  ;ALWAYS OUTPUT GLOBAL SYMBOL
              06   11  0414   694          BRB     10$
                     0416   695
                     0416   696 LBL1::                                       ;LABEL = ID DCOLON
 56   FFFC'CF47   D0  0416   697          MOVL    W^MAC$AL_VALSTACK-4[R7],R6 ;POINT TO ID SYMBOL BLOCK
                     041C   698 10$:
                     041C   699 LBL_X:                                       ;ENTRY FOR .ENTRY
 12 09 A6    06   E0  041C   700          BBS     #SYM$V_LOCAL,SYM$W_FLAG(R6),30$ ;BRANCH IF LOCAL SYMBOL
    03 0005'CF   E8  0421   701          BLBS    W^ENB$G_LOCALSYMB+SYM$L_VAL,20$ ;BRANCH IF ENABLE LSB
        FBD7'    30  0426   702          BSBW    MAC$SET_NEW_LSB             ;NO--MAKE A NEW LSB
    05 0005'CF   E9  0429   703 20$:     BLBC    W^ENB$G_DEBUG+SYM$L_VAL,30$ ;BRANCH IF NO ENABLE DEBUG
00 09 A6    05   E3  042E   704          BBCS    #SYM$V_DEBUG,SYM$W_FLAG(R6),30$ ;NO--TELL DEBUGGER ABOUT SYMBOL
08 09 A6    00   E1  0433   705 30$:     BBC     #SYM$V_DEF,SYM$W_FLAG(R6),40$ ;SYMBOL ALREADY DEFINED?
                     0438   706          $MAC_ERR MULDEFLBL                  ; Yes--send message
        FBC0'    30  043D   707          BSBW    MAC$ERRORPT                 ;
08 09 A6    03   E1  0440   708 40$:     BBC     #SYM$V_EXTRN,SYM$W_FLAG(R6),50$ ;IS SYMBOL EXTERNAL?
                     0445   709          $MAC_ERR SYMDCLEXTR                 ; Yes--send message
        FBB3'    30  044A   710          BSBW    MAC$ERRORPT
05 A6    0000'CF   D0  044D   711 50$:     MOVL    W^MAC$GL_PC,SYM$L_VAL(R6) ;SET SYMBOL VALUE
0C A6    0000'CF   90  0453   712          MOVB    W^MAC$GL_PSECT,SYM$B_SEG(R6) ;SET PSECT NUMBER OF SYMBOL
    09 A6    01   A8  0459   713          BISW2   #SYM$M_DEF,SYM$W_FLAG(R6) ;MARK AS DEFINED
 55  0000'CF   D0  045D   714          MOVL    W^MAC$GL_PSECTPTR,R5        ;POINT TO CURRENT PSECT
04 0D A5    03   E0  0462   715          BBS     #PSC$V_REL,PSC$W_OPTIONS(R5),60$ ;BRANCH IF RELOCATABLE
    09 A6    10   A8  0467   716          BISW2   #SYM$M_ABS,SYM$W_FLAG(R6) ;NO--FLAG SYMBOL AS ABSOLUTE
09 A5    0080 8F   A8  046B   717 60$:     BISW2   #SYM$M_REF,PSC$W_FLAG(R5) ;MARK PSECT AS REFERENCED
                     0471   718          $INTOUT_LW INT$_LGLAB,R6            ;OUTPUT COMMAND TO PASS 2
     55   00'8F   9A  0479   719          MOVZBL  #CRF$K_DEF,R5               ;SET DEFINITION
          FB80'   31  047D   720          BRW     MAC$CREF_SYM                ;CREF SYMBOL IF CROSS REFERENCING
                     0480   721
```

```
                0480   722          .DSABL  LSB
```

```
                     0480   724          .SBTTL  DATA GENERATION DIRECTIVES
                     0480   725
                     0480   726 ;++
                     0480   727 ; FUNCTIONAL DESCRIPTION:
                     0480   728 ;
                     0480   729 ;       BYTE/WORD/LONG/QUAD/SGNBYT/SGNWRD/OCTA ARE CALLED WHEN THE CORRESPONDING
                     0480   730 ;       DATA GENERATION DIRECTIVE IS SCANNED.  FLAGS ARE SET FOR THE
                     0480   731 ;       ROUTINES DALST2, DALST1, AND DATNUL TO PROCESS THE FOLLOWING
                     0480   732 ;       DATA ITEMS.
                     0480   733 ;
                     0480   734 ;--
                     0480   735
                     0480   736 BYTE::                                       ;DATA_TYPE = KBYTE
00     DD            0480   737          PUSHL   #0                          ;STACR INDEX
1F     10            0482   738          BSBB    DAT_COM                     ;GO TO COMMON ROUTINE
       01            0484   739          .BYTE   1                           ;1 BYTE PER ITEM
                     0485   740
                     0485   741 WORD::                                       ;DATA_TYPE = KWORD
01     DD            0485   742          PUSHL   #1                          ;STACR INDEX
1A     10            0487   743          BSBB    DAT_COM                     ;GO TO COMMON ROUTINE
       02            0489   744          .BYTE   2                           ;TWO BYTES PER ITEM
                     048A   745
                     048A   746 LONG::                                       ;DATA_TYPE = KLONG
02     DD            048A   747          PUSHL   #2                          ;STACR INDEX
15     10            048C   748          BSBB    DAT_COM                     ;GO TO COMMON ROUTINE
       04            048E   749          .BYTE   4                           ;FOUR BYTES PER ITEM
                     048F   750
                     048F   751 QUAD::                                       ;DATA_TYPE = KQUAD
03     DD            048F   752          PUSHL   #3                          ;STACR INDEX
10     10            0491   753          BSBB    DAT_COM                     ;GO TO COMMON ROUTINE
       08            0493   754          .BYTE   8                           ;EIGHT BYTES PER ITEM
                     0494   755
                     0494   756 SGNBYT::                                     ;DATA_TYPE = KSGNB
04     DD            0494   757          PUSHL   #4                          ;STACR INDEX
0B     10            0496   758          BSBB    DAT_COM                     ;GO TO COMMON ROUTINE
       01            0498   759          .BYTE   1                           ;ONE BYTE PER ITEM
                     0499   760
                     0499   761 SGNWRD::                                     ;DATA_TYPE = KSGNW
05     DD            0499   762          PUSHL   #5                          ;STACR INDEX
06     10            049B   763          BSBB    DAT_COM                     ;GO TO COMMON ROUTINE
       02            049D   764          .BYTE   2                           ;TWO BYTES PER ITEM
                     049E   765
                     049E   766 OCTA::                                       ;DATA_TYPE = KOCTA
06     DD            049E   767          PUSHL   #6                          ;STACR INDEX
01     10            04A0   768          BSBB    DAT_COM                     ;GOTO COMMON ROUTINE
       10            04A2   769          .BYTE   16                          ;SIXTEEN BYTES PER ITEM
                     04A3   770
                     04A3   771 DAT_COM:
0000'CF   9E   9A    04A3   772          MOVZBL  @(SP)+,W^MAC$GL_OPSIZE      ;STORE OPERAND SIZE
    0000'CF 8ED0     04A8   773          POPL    W^MAC$GL_DIRFLG             ;STORE INDEX
  00 6B   06   E3    04AD   774          BBCS    #FLG$V_EVALEXPR,(R11),.+1 ;ALLOW EXPRESSION EVALUATION
                     04B1   775 ;
                     04B1   776 ; CONTINUE ON INTO DATA_EXIT
                     04B1   777 ;
```

```
                                04B1    779 ;++
                                04B1    780 ; FUNCTIONAL DESCRIPTION:
                                04B1    781 ;
                                04B1    782 ;       'ADDRES' IS CALLED WHEN A .ADDRESS DIRECTIVE IS SCANNED.
                                04B1    783 ;       ALL THAT IS DONE IS TO SET FLAGS AND ENSURE THAT THERE
                                04B1    784 ;       IS ROOM IN THE INTERMEDIATE BUFFER TO CONTAIN THE EXPRESSION.
                                04B1    785 ;
                                04B1    786 ;--
                                04B1    787
                                04B1    788 ADDRES::                                        ;ADDR_TYPE = KADDRESS
                                04B1    789 DATA_EXIT:
        0000'CF    59   D1      04B1    790             CMPL    R9,W^MAC$GL_INTWRNPT        ;NEAR THE END OF THE BUFFER?
                     03   1B      04B6    791             BLEQU   10$                         ;IF LEQ NO
                FB45'     30      04B8    792             BSBW    MAC$OUTFRAME                ;YES--WRITE BUFFER OUT
        0000'CF    59   D0      04BB    793 10$:        MOVL    R9,W^MAC$GL_EXPPTR          ;SAVE START OF EXPRESSION
        0000'CF    59   D0      04C0    794             MOVL    R9,W^MAC$GL_EXPEND          ;AND END OF EXPRESSION
              0000'CF    D4      04C5    795             CLRL    W^MAC$GL_ABSFLAG            ;ASSUME ABSOLUTE EXPR
              0000'CF    D4      04C9    796             CLRL    W^MAC$GL_PRMSEG             ;ABSOLUTE SEGMENT
           00 6B    04   E5      04CD    797             BBCC    #FLG$V_DATRPT,(R11),.+1     ;NO REPEAT YET
    6B   00000084 8F    C8      04D1    798             BISL2   #FLG$M_EXPOPT!FLG$M_COMPEXPR,(R11) ;ALLOW EXPRESSION OPT.
                                04D8    799                                                 ;   AND ASSUME COMPILE TIME EXPR
                          05    0[illegible]    800             RSB
```

```
                     04D9    802 ;++
                     04D9    803 ; FUNCTIONAL DESCRIPTION:
                     04D9    804 ;
                     04D9    805 ;       'STOADR' IS CALLED FOR EACH ITEM FOUND IN A .ADDRESS DIRECTIVE.
                     04D9    806 ;       CODE IS PUT IN THE INTERMEDIATE BUFFER TO STACK THE VALUE,
                     04D9    807 ;       AND STORE POSITION INDPENDENT DATA.  FLAGS ARE THEN INITIALIZED
                     04D9    808 ;       FOR THE NEXT ITEM.
                     04D9    809 ;
                     04D9    810 ;--
                     04D9    811
                     04D9    812 STOADR::                                ;ADDR_LIST = EXPR ! ADDR_LIST DCOMMA EXPR
0000'CF   D5         04D9    813         TSTL    W^MAC$GL_ABSFLAG        ;ABSOLUTE EXPRESSION?
      0E   12         04DD    814         BNEQ    10$                     ;IF NEQ NO
 FB1E'     30         04DF    815         BSBW    MAC$OPTIMIZEXPR         ;YES--WIPE IT OUT
                      04E2    816         $INTOUT_LW INT$_STKL,<W^MAC$AL_VALSTACK[R7]> ;AND STACK THE VALUE
                      04ED    817 10$:    $INTOUT_X INT$_SPID             ;STORE PIC DATA
                      04F3    818         $INC_PC #4                      ;COUNT FOUR BYTES
 FFB6     31         04F8    819         BRW     DATA_EXIT               ;INIT FOR NEXT ADDRESS
                      04FB    820
                     04FB    821 ;++
                     04FB    822 ; FUNCTIONAL DESCRIPTION:
                     04FB    823 ;
                     04FB    824 ;       'DATARG' IS CALLED FOR EACH ITEM IN A BYTE/WORD/LONG/QUAD
                     04FB    825 ;       DIRECTIVE.  FLAGS ARE INITIALIZED FOR THE NEXT ITEM.
                     04FB    826 ;
                     04FB    827 ;--
                     04FB    828
                     04FB    829 DATARG::                                ;DATA_LIST = EXPR
                     04FB    830                                         ;DATA_LIST = DATA_LIST DCOMMA EXPR
0000'CF   D5         04FB    831         TSTL    W^MAC$GL_ABSFLAG        ;ABSOLUTE EXPRESSION?
      04   13         04FF    832         BEQL    10$                     ;IF EQL YES
00 6B    07   E5     0501    833         BBCC    #FLG$V_EXPOPT,(R11),10$ ;NO--NO OPTIMIZATION
                     0505    834 10$:
                     0505    835 ;
                     0505    836 ; THE FOLLOWING ALLOWS EVALUATION OF REPEAT COUNT
                     0505    837 ;
00 6B    02   E3     0505    838         BBCS    #FLG$V_COMPEXPR,(R11),.+1 ;ASSUME COMPILE TIME EXPRESSION
0000'CF   D4         0509    839         CLRL    W^MAC$GL_ABSFLAG        ;ASSUME ABSOLUTE
0000'CF   D4         050D    840         CLRL    W^MAC$GL_PRMSEG         ;ABS PSECT
00 6B    04   E5     0511    841         BBCC    #FLG$V_DATRPT,(R11),.+1 ;NO REPEAT COUNT YET
               05     0515    842         RSB
                     0516    843
                     0516    844 ;++
                     0516    845 ; FUNCTIONAL DESCRIPTION:
                     0516    846 ;
                     0516    847 ;       'DATNUL' IS CALLED WHEN A NULL DATA ITEM IS FOUND IN A
                     0516    848 ;       BYTE/WORD/LONG/QUAD/OCTA DIRECTIVE.  A ZERO VALUE IS EMITTED
                     0516    849 ;       TO PASS 2 AND FLAGS  ARE INITIALIZED FOR THE NEXT ITEM.
                     0516    850 ;
                     0516    851 ;--
                     0516    852
                     0516    853 DATNUL::                                ;DATA_STAT = DATA_TYPE <NULL>
55   0000'CF   D0     0516    854         MOVL    W^MAC$GL_DIRFLG,R5      ;GET INDEX FOR DATA TYPE
50   00000000'E5 9A   051B    855         MOVZBL  L^DAT_NUL_CMD(R5),R0    ;GET COMMAND
               00 DD   0522    856         PUSHL   #0                      ;STACK A 0
          FAD9' 30     0524    857         BSBW    MAC$INTOUT_1_LW         ;SEND TO INT. BUFFER
53   00000031'E5 9A   0527    858         MOVZBL  L^DAT_SHIFT_FACT(R5),R3 ; Get shift factor
```

```
                03   53   91   052E   859          CMPB    R3,#3                   ; Was this .QUAD or .OCTA?
                     1D   19   0531   860          BLSS    10$                     ; No if LSS
                               0533   861          $INTOUT_LW INT$_STIL,<#0>       ; Set bits 32-63 as zero
                04   53   91   053B   862          CMPB    R3,#4                   ; Was this .OCTA?
                     10   12   053E   863          BNEQ    10$                     ; No if NEQ
                               0540   864          $INTOUT_LW INT$_STIL,<#0>       ; Set bits 64-95 and
                               0548   865          $INTOUT_LW INT$_STIL,<#0>       ; bits 96-127 as zero
                               0550   866 10$:     $INC_PC W^MAC$G[_OPSIZE         ;COUNT THE BYTES
                    FF57   31  0557   867          BRW     DATA_EXIT               ;INIT FOR NEXT ITEM
```

```
                     055A   869 ;++
                     055A   870 ; FUNCTIONAL DESCRIPTION:
                     055A   871 ;
                     055A   872 ;       'DALST2' AND 'DALST1' ARE CALLED TO PROCESS THE ITEMS IN
                     055A   873 ;       A DATA-LIST FOR BYTE/WORD/LONG/QUAD/OCTA DIRECTIVES.  'DALST2'
                     055A   874 ;       IS CALLED IF THIS IS A REPEAT ITEM, AND 'DALST1' IS CALLED
                     055A   875 ;       IF IT IS NOT.
                     055A   876 ;
                     055A   877 ;--
                     055A   878
                     055A   879 DALST2::                                     ;DATA_ARGS = DATA_LIST DSQOPN EXPR DSQCLS
      6B   04   E3   055A   880          BBCS    #FLG$V_DATRPT,(R11),-       ;THIS IS REPEATED DATA
           03        055D   881                  DALST1
                     055E   882 QUDSTR::                                     ;DATA_STAT = QUAD_HEAD PRIMITIVE
                     055E   883 OCTSTR::                                     ;DATA_STAT = OCTA_HEAD PRIMITIVE
        FF9A   30    055E   884          BSBW    DATARG                      ;INIT DATA FLAGS
                     0561   885 DALST1::                                     ;DATA_ARGS = DATA_LIST
55   0000'CF   D0    0561   886          MOVL    W^MAC$GL_DIRFLG,R5          ;GET DATA TYPE INDEX
  2D 6B   04   E1    0566   887          BBC     #FLG$V_DATRPT,(R11),30$     ;BRANCH IF NOT REPEAT
                     056A   888 ;
                     056A   889 ; THIS IS REPEATED DATA TYPE
                     056A   890 ;
        0000'CF   D5    056A   891          TSTL    W^MAC$GL_ABSFLAG            ;IS REPEAT COUNT ABSOLUTE?
           08   12    056E   892          BNEQ    10$                         ;IF NEQ NO--ERROR
50   FFFC'CF47   D0    0570   893          MOVL    W^MAC$AL_VALSTACK-4[R7],R0 ;YES--GET REPEAT COUNT
           0D   11    0576   894          BRB     20$                         ;AND SKIP AHEAD
                     0578   895 10$:    $MAC_ERR RPTCNTNABS                 ; No--get error code
        FA80'   30    057D   896          BSBW    MAC$ERRORPT                 ;ISSUE MESSAGE TO PASS 2
     FFFC'CF47   D4    0580   897          CLRL    W^MAC$AL_VALSTACK-4[R7]     ;DO NO REPEATING
50 00000007'E5   9A    0585   898 20$:    MOVZBL  L^DAT_RPT_CMD(R5),R0        ;GET COMMAND
        FA71'   30    058C   899          BSBW    MAC$INTOUT_X                ;ISSUE TO PASS 2
50   FFFC'CF47   D0    058F   900          MOVL    W^MAC$AL_VALSTACK-4[R7],R0 ;GET THE REPEAT COUNT
           36   11    0595   901          BRB     60$                         ;FINISH UP
                     0597   902 ;
                     0597   903 ; NOT A REPEAT
                     0597   904 ;
     25 6B   07   E1    0597   905 30$:    BBC     #FLG$V_EXPOPT,(R11),40$ ;BRANCH IF NOT OPTIMIZABLE
        FA62'   30    059B   906          BSBW    MAC$OPTIMIZEXPR             ;YES--WIPE OUT EXPRESSION
     0000'CF47   DD    059E   907          PUSHL   W^MAC$AL_VALSTACK[R7]       ;STACK THE VALUE
50 00000015'EF45   D0    05A3   908          MOVL    L^DAT_TRUNC_CHK[R5],R0      ;GET TRUNCATION ROUTINE CHECK ADDRESS
           02   13    05AB   909          BEQL    33$                         ;IF EQL NO NEED TO CHECK
           60   16    05AD   910          JSB     (R0)                        ;CHECK FOR TRUNCATION AND REPORT ERROR
55   0000'CF   D0    05AF   911 33$:    MOVL    W^MAC$GL_DIRFLG,R5          ;RETRIEVE DATA TYPE INDEX AGAIN
50 00000000'E5   9A    05B4   912          MOVZBL  L^DAT_NUL_CMD(R5),R0        ;GET THE COMMAND
        FA42'   30    05BB   913 35$:    BSBW    MAC$INTOUT_1_LW             ;SEND TO INT. FILE
           0A   11    05BE   914          BRB     50$                         ;CONTINUE
                     05C0   915 ;
                     05C0   916 ; NOT OPTIMIZED, NOT REPEATED
                     05C0   917 ;
50 0000000E'E5   9A    05C0   918 40$:    MOVZBL  L^DAT_STO_CMD(R5),R0        ;GET COMMAND
        FA36'   30    05C7   919          BSBW    MAC$INTOUT_X                ;SEND TO INT. FILE
        50   01   9A    05CA   920 50$:    MOVZBL  #1,R0                       ;USE REPEAT COUNT OF 1
                     05CD   921 ;
                     05CD   922 ; FINISH UP
                     05CD   923 ;
53 00000031'E5   9A    05CD   924 60$:    MOVZBL  L^DAT_SHIFT_FACT(R5),R3 ; Get shift factor
   50   50   53   78    05D4   925          ASHL    R3,R0,R0                    ; Figure total allocation
```

```
                 05D8   926          $INC_PC R0                          ;COUNT IN PASS 1
03   53   91     05DD   927          CMPB    R3,#3                       ; Was this .QUAD or .OCTA
     24   19     05E0   928          BLSS    70$                         ; No if LSS
                 05E2   929          $INTOUT_LW INT$_STIL,<W^MAC$GL_VAL3> ; Send bits 32-63 to intermediate file
04   53   91     05EC   930          CMPB    R3,#4                       ; Was this .OCTA?
     14   12     05EF   931          BNEQ    65$                         ; No if NEQ
                 05F1   932          $INTOUT_LW INT$_STIL,<W^MAC$GQ_VAL2+0> ; Send bits 64-95 and then
                 05FB   933          $INTOUT_LW INT$_STIL,<W^MAC$GQ_VAL2+4> ; bits 96-127 to intermediate file
                 0605   934 65$:
          05     0605   935          RSB
     FEA8 31     0606   936 70$:     BRW     DATA_EXIT                   ;INIT FOR NEXT ELEMENT
```

```
                     0609   938          .SBTTL  ENTRY POINT DEFINITION DIRECTIVES
                     0609   939
                     0609   940 ;++
                     0609   941 ; FUNCTIONAL DESCRIPTION:
                     0609   942 ;
                     0609   943 ;       VECTRO IS CALLED WHEN A .VECTOR DIRECTIVE WITH NO EPT MASK
                     0609   944 ;       IS SCANNED.
                     0609   945 ;
                     0609   946 ;--
                     0609   947
                     0609   948 VECTRO::                                    ;DIRECTIVE = KVECTOR ID
                     0609   949          $INTOUT_LW INT$_STKEPT,<W^MAC$AL_VALSTACK[R7]> ;STACK ENTRY POINT MASK
                     0614   950          $INTOUT_X INT$_STOW                ; Store word
           69   11   061A   951          BRB     ENTRY_VEC_XIT              ;TAKE COMMON EXIT
                     061C   952
                     061C   953 ;++
                     061C   954 ; FUNCTIONAL DESCRIPTION:
                     061C   955 ;
                     061C   956 ;       VECTR2 AND VECTR1 ARE CALLED WHEN .VECTOR DIRECTIVES ARE
                     061C   957 ;       SCANNED WITH AN EPT MASK.  CODE IS EMITTED TO STACK THE
                     061C   958 ;       EPT AND OR IT WITH THE EXPRESSION ON THE STACK.
                     061C   959 ;
                     061C   960 ;--
                     061C   961
                     061C   962 VECTR2::                                    ;DIRECTIVE = KVECTOR ID EXPR
     52  FFFC'CF47   D0  061C   963          MOVL    W^MAC$AL_VALSTACK-4[R7],R2 ;POINT TO SYMBOL
           06   11   0622   964          BRB     VEC_COM                    ;
                     0624   965
                     0624   966 VECTR1::                                    ;DIRECTIVE = KVECTOR ID DCOMMA EXPR
     52  FFF8'CF47   D0  0624   967          MOVL    W^MAC$AL_VALSTACK-8[R7],R2 ;POINT TO SYMBOL
                     062A   968 VEC_COM:
                     062A   969          $INTOUT_LW INT$_STKEPT,R2          ;STACK EPT
                     0632   970          $INTOUT_X INT$_OR                  ;OR WITH EXPR ON STACK
                     0638   971          $INTOUT_X INT$_STOW                ; Store word
           24   11   063E   972          BRB     ENTRY_VECTOR               ;
                     0640   973
                     0640   974 ;++
                     0640   975 ; FUNCTIONAL DESCRIPTION:
                     0640   976 ;
                     0640   977 ;       ENTRY1 AND ENTRY2 ARE CALLED TO PROCESS .ENTRY DIRECTIVES.  THE
                     0640   978 ;       ONLY DIFFERENCE BETWEEN THEM IS THAT ENTRY1 IS CALLED IF THERE
                     0640   979 ;       WAS A COMMA BETWEEN THE ID AND THE EXPRESSIION AND ENTRY2 IS
                     0640   980 ;       CALLED IF THERE WAS NO COMMA.
                     0640   981 ;
                     0640   982 ;--
                     0640   983
                     0640   984 ENTRY1::                                    ;DIRECTIVE = KENTRY ID DCOMMA EXPR
     56  FFF8'CF47   D0  0640   985          MOVL    W^MAC$AL_VALSTACK-8[R7],R6 ;POINT TO SYMBOL BLOCK
           06   11   0646   986          BRB     ENTRY_COM
                     0648   987
                     0648   988 ENTRY2::                                    ;DIRECTIVE = KENTRY ID EXPR
     56  FFFC'CF47   D0  0648   989          MOVL    W^MAC$AL_VALSTACK-4[R7],R6 ;POINT TO SYMBOL BLOCK
                     064E   990 ENTRY_COM:
       0204 8F   A8  064E   991          BISW2   #SYM$M_EPT!SYM$M_GLOBL,- ;MARK AS GLOBAL EPT
         09 A6       0652   992                  SYM$W_FLAG(R6)  ;...
           FDC5   30 0654   993          BSBW    LBL_X                      ;DEFINE LABEL
                     0657   994          $INTOUT_LW INT$_EPT,<R6,W^MAC$AL_VALSTACK[R7]> ;PROCESS EPT ON PASS 2
```

```
                                0664   995 ENTRY_VECTOR:
                     0000'CF    D5  0664   996         TSTL    W^MAC$GL_ABSFLAG          ;ABSOLUTE EXPR?
                           13   12  0668   997         BNEQ    10$                       ;IF NEQ NO
0000'CF47    00003003      8F    D3  066A   998         BITL    #^X3003,W^MAC$AL_VALSTACK[R7] ;YES--ANY ILLEGAL BITS SET?
                           0F   13  0674   999         BEQL    20$                       ;IF EQL NO
                                0676  1000         $MAC_ERR ILLMASKBIT                ; Yes--get message code
                           05   11  0679  1001         BRB     15$                       ;
                                067B  1002 10$:      $MAC_ERR EMSKNOTABS               ; Entry mask not absolute
                       F97B'    30  0680  1003 15$:      BSBW    MAC$ERRORPT               ;REPORT TO PASS 2
                                0683  1004 20$:
                                0683  1005 ENTRY_VEC_XIT:
                                0683  1006         $INC_PC #2                        ;COUNT TWO BYTES
          50  00000000'EF       D0  0688  1007         MOVL    MAC$GL_PSECTPTR, R0       ;AND MARK THE PSECT AS REFERENCED.
           09 A0    0080 8F      A8  0689  1008         BISW2   #SYM$M_REF,PSC$W_FLAG(R0)
                           05   11  0695  1009         RSB
                                0696  1010
                                0696  1011 ;++
                                0696  1012 ; FUNCTIONAL DESCRIPTION:
                                0696  1013 ;
                                0696  1014 ;       THIS ROUTINE IS CALLED TO PROCESS THE .TRANSFER DIRECTIVE.
                                0696  1015 ;       CODE IS EMITTED TO PASS 2 TO SEND A REDEFINITION COMMAND
                                0696  1016 ;       TO THE LINKER.
                                0696  1017 ;
                                0696  1018 ;--
                                0696  1019
                                0696  1020 XFER::                                     ;DIRECTIVE = KXFER ID
                                0696  1021         $INTOUT_LW INT$_REDEF,<W^MAC$AL_VALSTACK[R7]>  ;TELL PASS 2
          50  00000000'EF       D0  06A1  1022         MOVL    MAC$GL_PSECTPTR, R0       ;AND MARK THE PSECT AS REFER
            01    0C A0        91  06A8  1023         CMPB    PSC$B_SEG(R0), #1         ;ARE WE DEALING WITH
                           06   12  06AC  1024         BNEQ    10$                       ;THE BLANK PSECT?
           09 A0    0080 8F      A8  06AE  1025         BISW2   #SYM$M_REF,PSC$W_FLAG(R0) ;YES MARK IT AS REFERENCED.
                           05   11  06B4  1026 10$:      RSB
                                06B5  1027
                                06B5  1028         .END
```

```
$COUNT            =  0000003B
AB                =  00000001
AD                =  0000C008
ADDRES               000004B1 RG     04
ADM$_ABSOLUTE     =  00000002
ADM$_BYTE_DISP    =  0000000A
ADM$_DFBYTEDISP=  0000000B
ADM$_DFLONGDISP=  0000000F
ADM$_DFRAUTOINC=  00000009
ADM$_DFWORDDISP=  0000000D
ADM$_IMMEDIATE    =  00000001
ADM$_INDEX        =  00000004
ADM$_LITERAL      =  00000000
ADM$_LONG_DISP    =  0000000E
ADM$_MAXMOD       =  0000000F
ADM$_PIC          =  0C000003
ADM$_REGAUTODEC=  00000007
ADM$_REGAUTOINC=  00000008
ADM$_REGISTER     =  00000005
ADM$_RRIND        =  00000006
ADM$_WORD_DISP    =  0000000C
AF                =  00008004
AG                =  0000A008
AH                =  00009010
AL                =  00000004
AO                =  00000010
AQ                =  00000008
ARG$K_SIZE        =  000003E8
ASSGNT               00000285 RG     04
ASSHD1               0000023C RG     04
ASSHD2               0000022A RG     04
ASSHD3               00000224 RG     04
ASSIGN_HEAD          00000242 R      04
AUD$K_SIZE        =  00000010
AW                =  00000002
B                 =  00000001
BLKBYT               0000039A RG     04
BLKLNG               000003A0 RG     04
BLKOCT               000003A6 RG     04
BLKQUD               000003A3 RG     04
BLKWRD               0000039D RG     04
BLNK              =  00000020
BSTAT1               000003BA RG     04
BSTAT2               000003C5 RG     04
BYTE                 00000480 RG     04
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
CR                =  0000000D
CRF$K_DEF            ********    X   04
D                 =  0000C008
DALST1               00000561 RG     04
DALST2               0000055A RG     04
DATARG               000004FB RG     04
DATA_EXIT            000004B1 R      04
DATNUL               00000516 RG     04
DAT_COM              000004A3 R      04
DAT_NUL_CMD          00000000 R      03
DAT_RPT_CMD          00000007 R      03
DAT_SHIFT_FACT       00000031 R      03
DAT_STO_CMD          0000000E R      03
DAT_TRUNC_CHK        00000015 R      03
ENB$G_DEBUG          ********      X 04
ENB$G_LOCALSYMB      ********      X 04
ENB$G_SUPPRESS       ********      X 04
ENTRY1               00000640 RG     04
ENTRY2               00000648 RG     04
ENTRY_COM            0000064E R      04
ENTRY_VECTOR         00000664 R      04
ENTRY_VEC_XIT        00000685 R      04
ERR               =  00000000
F                 =  00008004
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
G               = 0000A008
H               = 00009010
HASHSZ          = 0000007F
HYPHEN          = 0000002D
INP$K_BUFSIZ    = 000003E8
INT$K_BUFSIZ    = 000013F4
INT$K_BUFWRN    = 00001390
INT$_ADD        = 00000001
INT$_AND        = 00000002
INT$_ASH        = 00000003
INT$_ASN        = 0000000C
INT$_AUGPC      = 0000000D
INT$_BDST       = 0000000E
INT$_CHKL       = 0000000F
INT$_DIV        = 00000004
INT$_END        = 00000010
INT$_EPT        = 00000011
INT$_ERR        = 00000012
INT$_ETX        = 00000013
INT$_FNEWL      = 00000014
INT$_ILG        = 00000000
INT$_INFO       = 0000003A
INT$_LGLAB      = 00000015
INT$_MACL       = 00000016
INT$_MUL        = 00000005
INT$_NEG        = 00000006
INT$_NEWL       = 00000017
INT$_NEWP       = 00000018
INT$_NOT        = 00000007
INT$_OP         = 00000019
INT$_OR         = 00000008
INT$_PRIL       = 0000001A
INT$_PRT        = 0000001B
INT$_PSECT      = 0000001C
INT$_REDEF      = 0000001D
INT$_REF        = 0000001E
INT$_REST       = 0000001F
INT$_SAME       = 00000009
INT$_SAVE       = 00000020
INT$_SBTTL      = 00000021
INT$_SETFLAG    = 00000022
INT$_SETLONG    = 00000023
INT$_SPIC       = 00000024
INT$_SPID       = 00000025
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
L               = 00000004
LBL1              00000416 RG    04
LBL2              00000404 RG    04
LBL_X             0000041C R     04
LONG              0000048A RG    04
LST$K_BUFSIZ    = 00000086
LST$K_L_P_PAGE  = 0000003C
LST$K_TITLE_SIZ= 00000028
MAC$AB_LINEBF     ********    X  04
MAC$AL_VALSTACK   ********    X  04
MAC$AW_ILLMODTB   ********    X  04
MAC$CK_BYT_TRU1   ********    X  03
MAC$CK_SBY_TRU1   ********    X  03
MAC$CK_SWD_TRU1   ********    X  03
MAC$CK_WRD_TRU1   ********    X  03
MAC$CREF_OPCODE   ********    X  04
MAC$CREF_SYM      ********    X  04
MAC$ERRORPT       ********    X  04
MAC$ERRORPX       ********    X  04
MAC$GB_MODE       ********    X  04
MAC$GB_RDXNDX     ********    X  04
MAC$GB_REG        ********    X  04
MAC$GB_VAL3       ********    X  04
MAC$GL_ABSFLAG    ********    X  04
MAC$GL_ASNPTR     ********    X  04
MAC$GL_DIRFLG     ********    X  04
MAC$GL_ERRPTX     ********    X  04
MAC$GL_EXPEND     ********    X  04
MAC$GL_EXPOPVL1   ********    X  04
MAC$GL_EXPPTR     ********    X  04
MAC$GL_HIGH_32    ********    X  04
MAC$GL_INTWRNPT   ********    X  04
MAC$GL_LINBAS     ********    X  04
MAC$GL_LINENUM    ********    X  04
MAC$GL_MOPNUM     ********    X  04
MAC$GL_MOPPTR     ********    X  04
MAC$GL_OPSIZE     ********    X  04
MAC$GL_PC         ********    X  04
MAC$GL_PRMSEG     ********    X  04
MAC$GL_PSECT      ********    X  04
MAC$GL_PSECTPTR   ********    X  04
MAC$GL_RECHDBUF   ********    X  04
MAC$GL_SAVE_PC    ********    X  04
MAC$GL_SAV_BAS    ********    X  04
MAC$GL_SAV_LIN    ********    X  04
MAC$GL_SAV_PAG    ********    X  04
MAC$GL_SRCPAG     ********    X  04
MAC$GL_VAL3       ********    X  04
MAC$GL_VALUE      ********    X  04
MAC$GQ_HIGH_64    ********    X  04
MAC$GQ_VAL2       ********    X  04
MAC$INTOUT_1_LW   ********    X  04
MAC$INTOUT_2_LW   ********    X  04
MAC$INTOUT_ASN    0000035B RG    04
MAC$INTOUT_N      ********     X  04
MAC$INTOUT_WD     ********     X  04
MAC$INTOUT_X      ********     X  04
MAC$MUL_DEF_CHK   0000037C RG    04
MAC$OPTIMIZEXPR   ********     X  04
MAC$OUTFRAME      ********    X  04
MAC$SET_NEW_LSB   ********    X  04
MAC$SET_PC        ********    X  04
MAC$_ASGNMNTSYN= 007D9022
MAC$_BLKEXPNABS= 007D904A
MAC$_EMSKNOTABS= 007D9072
MAC$_ILLBRDEST = 007D90BA
MAC$_ILLMASKBIT= 007D90FA
MAC$_ILLMODE   = 007D9102
MAC$_MULDEFLBL = 007D915A
MAC$_NOTENUFOPR= 007D917A
MAC$_OPRNDSYNX = 007D91A2
MAC$_RPTCNTNABS= 007D91D2
MAC$_SYMDCLEXTR= 007D91DA
MAC$_TOOMNYOPND= 007D9202
MACH_OP_EXIT      0000003A R     04
MAC_SUBSYS      = 0000007D
MB              = 00000041
MD              = 0000C048
MF              = 00008044
MG              = 0000A048
MH              = 00009050
MINST1            0000000F RG    04
ML              = 00000044
MO              = 00000050
MQ              = 00000048
MW              = 00000042
O               = 00000010
```

```
OBJ$K_BUFSIZ    =  00000200
OCTA               0000049E  RG    04
OCTSTR             0000055E  RG    04
OPD$M_ADDR      =  00000000
OPD$M_BB        =  000000A1
OPD$M_BW        =  000000C2
OPD$M_D_FLOAT   =  0000C000
OPD$M_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_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
OPRAND             0000008C  RG    04
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$W_OPTIONS      0000000D
Q               =  00000008
QUAD               0000048F  RG    04
QUDSTR             0000055E  RG    04
RB              =  00000021
RD              =  0000C028
RDX$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
RW              =  00000022
SEMI            =  0000003B
SGNBYT             00000494  RG    04
SGNWRD             00000499  RG    04
STAT1              00000000  RG    04
STB$K_PG_MISS   =  0000000A
STOADR             000004D9  RG    04
SYM$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
VECTR0             00000609  RG    04
VECTR1             00000624  RG    04
VECTR2             0000061C  RG    04
VEC_COM            0000062A  R     04
VF              =  00008084
VG              =  0000A088
VH              =  00009090
VL              =  00000084
VO              =  00000090
VQ              =  00000088
VW              =  00000082
W               =  00000002
WB              =  00000061
WD              =  0000C068
WF              =  00008064
WG              =  0000A068
WH              =  00009070
WL              =  00000064
WO              =  00000070
WORD               00000485  RG    04
```

```
WQ              = 00000068
WW              = 00000062
X               = 00000010
X1              = 00000033
X2              = 00080000
XFER              00000698 RG    04
```

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$RO_DATA	00000038 (56.)	03 (3.)	NOPIC USR CON REL GBL NOSHR NOEXE RD NOWRT NOVEC LONG
MAC$RO_CODE_P1	000006B7 (1719.)	04 (4.)	NOPIC USR CON REL GBL NOSHR EXE RD NOWRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	29	00:00:00.02	00:00:02.02
Command processing	103	00:00:00.36	00:00:03.48
Pass 1	259	00:00:05.02	00:00:25.63
Symbol table sort	0	00:00:00.60	00:00:02.90
Pass 2	196	00:00:01.77	00:00:06.58
Symbol table output	43	00:00:00.22	00:00:01.00
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	634	00:00:08.01	00:00:41.63

```
The working set limit was 1350 pages.
48829 bytes (96 pages) of virtual memory were used to buffer the intermediate code.
There were 40 pages of symbol table space allocated to hold 587 non-local and 67 local symbols.
1028 source lines were read in Pass 1, producing 29 object records in Pass 2.
21 pages of virtual memory were used to define 17 macros.
```

Macro library statistics

Macro library name	Macros defined
_255DUA28:[MACRO.OBJ]MACRO.MLB;1	15
_255DUA28:[SYSLIB]STARLET.MLB;2	3
TOTALS (all libraries)	18

```
625 GETS were required to define 18 macros.

There were no errors, warnings or information messages.

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

ACTPRI LIS

ARGSCN LIS

BDYSCN LIS

CRFSUB LIS

ACTOPC LIS

ACTSTA LIS

APSECT LIS

CRFDAT LIS

ACTREF LIS

COMPUT LIS