M 5

```
MMM         MMM     AAAAAAAAA        CCCCCCCCCCCC  RRRRRRRRRRRR        OOOOOOOOO
MMM         MMM     AAAAAAAAA        CCCCCCCCCCCC  RRRRRRRRRRRR        OOOOOOOOO
MMM         MMM     AAAAAAAAA        CCCCCCCCCCCC  RRRRRRRRRRRR        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              RRRRRRRRRRRR     OOO         OOO
MMM         MMM  AAA         AAA  CCC              RRRRRRRRRRRR     OOO         OOO
MMM         MMM  AAA         AAA  CCC              RRRRRRRRRRRR     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     CCCCCCCCCCCC  RRR         RRR     OOOOOOOOO
MMM         MMM  AAA         AAA     CCCCCCCCCCCC  RRR         RRR     OOOOOOOOO
MMM         MMM  AAA         AAA     CCCCCCCCCCCC  RRR         RRR     OOOOOOOOO
```

```
_$
Sy
---
MA
MA
MA
MA
MA
MA
MA
MA
MA
MA
MA
MA
MA
MA
MA
MA
MA
MA
MA
MA
MA
MA
MA
MA
MA
MA
MA
MA
MA
MA
MA
MA
MA

MA
MA
MA
MA

MA
MA
MA
MA
MA
MA
MA
MA
MA
MA
MA
MA
```

```
**FILE**ID**ACTIF                                                   I 14                                    MA
                                                                                                            VO





  AAAAAA      CCCCCCCC  TTTTTTTTTT    IIIIII    FFFFFFFFFF
  AAAAAA      CCCCCCCC  TTTTTTTTTT    IIIIII    FFFFFFFFFF
AA      AA  CC              TT          II      FF
AA      AA  CC              TT          II      FF
AA      AA  CC              TT          II      FF
AA      AA  CC              TT          II      FF
AA      AA  CC              TT          II      FFFFFFFF
AA      AA  CC              TT          II      FFFFFFFF
AAAAAAAAAA  CC              TT          II      FF
AAAAAAAAAA  CC              TT          II      FF
AA      AA  CC              TT          II      FF              ....
AA      AA  CC              TT          II      FF              ....
AA      AA    CCCCCCCC      TT        IIIIII    FF              ....
AA      AA    CCCCCCCC      TT        IIIIII    FF              ....

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

MA
V0

```
0000            .TITLE  MACSACTIF        CONDITIONAL STATEMENT PROCESSOR
0000            .IDENT  'V04-000'
0000
0000    ;
0000    ;****************************************************************************
0000    ;*                                                                          *
0000    ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000    ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000    ;*  ALL RIGHTS RESERVED.                                                    *
0000    ;*                                                                          *
0000    ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000    ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000    ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000    ;*  TRANSFERRED.                                                            *
0000    ;*                                                                          *
0000    ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000    ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    ;*  CORPORATION.                                                            *
0000    ;*                                                                          *
0000    ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000    ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000    ;*                                                                          *
0000    ;*                                                                          *
0000    ;****************************************************************************
0000    ;
0000
0000    ;++
0000    ; FACILITY:     VAX MACRO ASSEMBLER OBJECT LIBRARY
0000    ;
0000    ; ABSTRACT:
0000    ;
0000    ; The VAX-11 MACRO assembler translates MACRO-32 source code into object
0000    ; modules for input to the VAX-11 LINKER.
0000    ;
0000    ; ENVIRONMENT:  USER MODE
0000    ;
0000    ; AUTHOR: Benn Schreiber, CREATION DATE: 20-AUG-78
0000    ;
0000    ; MODIFIED BY:
0000    ;
0000    ;       V03-001 MTR0027         Mike Rhodes             28-Feb-1983
0000    ;               Reset the expression evaluation flag after processing
0000    ;               an immediate if statement (.IIF).
0000    ;
0000    ;       V02.06  CNH0040         Chris Hume              15-Oct-1980
0000    ;               .ENDC ignored after local label in conditional suppressed
0000    ;               code.  (SCANER.MAR 02.14)
0000    ;
0000    ;       V01.05  RN0023          R. Newland              2-Nov-1979
0000    ;               New message codes to get error message from system
0000    ;               message file.
0000    ;
0000    ;       V01.05  RN0018          R. Newland              20-Oct-1979
0000    ;               Get arguments of .IF_IDENTICAL/.IF_DIFFERENT upper cased
0000    ;               before making comparison.
```

```
0000    58 ;
0000    59 ;       V01.04  RN0011          R. Newland              26-Sep-1979
0000    60 ;               New librarian support - remove truncation error
0000    61 ;
0000    62 ;       V01.03  RN0010          R. Newland               5-Sep-1979
0000    63 ;               Multipage IF arguments
0000    64 ;
0000    65 ;       V01.02  RN0005          R. Newland              14-Aug-1979
0000    66 ;               Variable symbol storage and remove .ALIGN LONG statements
0000    67 ;
0000    68 ;--
```

```
                 0000          .SBTTL  DECLARATIONS
                 0000   ;
                 0000   ; INCLUDE FILES:
                 0000   ;
                 0000
                 0000   ;
                 0000   ; MACROS:
                 0000   ;
                 0000
                 0000          $MAC_CTLFLGDEF                  ;DEFINE CONTROL FLAGS
                 0000          $MAC_GENVALDEF                  ;DEFINE GENERAL VALUES
                 0000          $MAC_INTCODDEF                  ;DEFINE INT. CODES
                 0000          $MAC_SYMBLKDEF                  ;DEFINE SYMBOL BLOCK OFFSETS
                 0000          $MAC_MNBDEF                     ; Define MXB offsets
                 0008          $MACMSGDEF                      ; Define message codes
                 0008
                 0008   ;
                 0008   ; EQUATED SYMBOLS:
                 0008   ;
                 0008
                 0008   ;
                 0008   ; OWN STORAGE:
                 0008   ;
                 0008
             00000000          .PSECT  MAC$RO_DATA,NOEXE,NOWRT,GBL,LONG
                 0000
                 0000   ;++
                 0000   ;      THESE ARE THE .IF CONDITION NAMES.  THE VALUE IS THE NAME OF
                 0000   ;      THE ROUTINE TO CALL.  IF THE ADDRESS HAS BIT 31 SET, THEN
                 0000   ;      THE ROUTINE MUST EVALUATE ITS OWN CONDITION, RATHER THAN
                 0000   ;      LETTING THE PARSER DO IT.
                 0000   ;
                 0000   ;--
                 0000
80000000         0000          IF_SPECIAL      =       ^X80000000      ;HIGH BIT IF SPECIAL
00000000         0000          INSYMP  =       0
                 0000
                 0000          $MAC_INSERT_SYX EQ,     IF_EQUAL                ;EQUAL TO ZERO
                 000C          $MAC_INSERT_SYX EQUAL,  IF_EQUAL                ;EQUAL TO ZERO
                 001B          $MAC_INSERT_SYX NE,     IF_NOT_EQUAL            ;NOT EQUAL TO ZERO
                 0027          $MAC_INSERT_SYX NOT_EQUAL,IF_NOT_EQUAL          ;NOT EQUAL TO ZERO
                 003A          $MAC_INSERT_SYX GT,     IF_GREATER              ;GREATER THAN ZERO
                 0046          $MAC_INSERT_SYX GREATER,IF_GREATER              ;GREATER THAN ZERO
                 0057          $MAC_INSERT_SYX LE,     IF_LESS_EQUAL           ;LESS THAN OR EQUAL ZERO
                 0063          $MAC_INSERT_SYX LESS_EQUAL,IF_LESS_EQUAL        ;LESS THAN OR EQUAL ZERO
                 0077          $MAC_INSERT_SYX GE,     IF_GTR_EQUAL            ;GREATER THAN OR EQUAL ZERO
                 0083          $MAC_INSERT_SYX GREATER_EQUAL,IF_GTR_EQUAL      ;GREATER THAN OR EQUAL ZERO
                 009A          $MAC_INSERT_SYX LT,     IF_LESS_THAN            ;LESS THAN ZERO
                 00A6          $MAC_INSERT_SYX LESS_THAN,IF_LESS_THAN          ;LESS THAN ZERO
                 00B9          $MAC_INSERT_SYX DF,     IF_DEFINED!IF_SPECIAL   ;DEFINED
                 00C5          $MAC_INSERT_SYX DEFINED,IF_DEFINED!IF_SPECIAL   ;DEFINED
                 00D6          $MAC_INSERT_SYX NDF,    IF_NOT_DEFINED!IF_SPECIAL;NOT DEFINED
                 00E3          $MAC_INSERT_SYX NOT_DEFINED,IF_NOT_DEFINED!IF_SPECIAL ;NOT DEFINED
                 00F8          $MAC_INSERT_SYX B,      IF_BLANK!IF_SPECIAL     ;BLANK
                 0103          $MAC_INSERT_SYX BLANK,  IF_BLANK!IF_SPECIAL     ;BLANK
                 0112          $MAC_INSERT_SYX NB,     IF_NOT_BLANK!IF_SPECIAL ;NOT BLANK
                 011E          $MAC_INSERT_SYX NOT_BLANK,IF_NOT_BLANK.IF_SPECIAL ;NOT BLANK
                 0131           $MAC_INSERT_SYX IDN,      IF_IDENTICAL!IF_SPECIAL ;IDENTICAL
                 013E           $MAC_INSERT_SYX IDENTICAL,IF_IDENTICAL!IF_SPECIAL ;IDENTICAL
                 0151           $MAC_INSERT_SYX DIF,      IF_DIFFERENT!IF_SPECIAL ;DIFFERENT
                 015E           $MAC_INSERT_SYX DIFFERENT,IF_DIFFERENT!IF_SPECIAL,-; DIFFERENT
                 015E                           IF_COND_NAMES
                 0171
                 0171   ;++
                 0171   ;
                 0171   ;       SPECIAL KEYWORDS WHICH ARE SCANNED WHILE PROCESSING THE
                 0171   ;       FALSE PART OF A CONDITIONAL ASSEMBLY.  THE VALUE IS A
                 0171   ;       POINTER TO A ROUTINE TO CALL WHEN THE KEYWORD IS DETECTED.
                 0171   ;
                 0171   ;--
                 0171
      00000000   0171           INSYMP  =       0                       ;START NEW LIST
                 0171
                 0171           $MAC_INSERT_SYX .END,     IF_ERROR              ;ERROR IF THIS SEEN
                 017F           $MAC_INSERT_SYX .IF,      IF_IN_AN_IF           ;.IF WITHIN AN IF
                 018C           $MAC_INSERT_SYX .IFF,     IFF                   ;.IFF
                 019A           $MAC_INSERT_SYX .IFT,     IFT                   ;.IFT
                 01A8           $MAC_INSERT_SYX .IFTF,    IFTF                  ;.IFTF
                 01B7           $MAC_INSERT_SYX .IF_FALSE, IFF                  ;.IF_FALSE
                 01CA           $MAC_INSERT_SYX .IF_TRUE,  IFT                  ;.IF_TRUE
                 01DC           $MAC_INSERT_SYX .IF_TRUE_FALSE, IFTF            ;.IF_TRUE_FALSE
                 01F4           $MAC_INSERT_SYX .ENDC,    ENDC,   IF_SPL_KEYWORDS
                 0203
            00000000            .PSECT  MAC$RO_CODE_P1,NOWRT,GBL,LONG
```

```
                              0000          .SBTTL  IFHD1 CONDITIONAL ASSEMBLY PROCESSOR
                              0000
                              0000     ;++
                              0000     ; FUNCTIONAL DESCRIPTION:
                              0000     ;
                              0000     ;        'IFHD1' IS CALLED WHEN A '.IF' CONDITIONAL ASSEMBLY IS
                              0000     ;        DETECTED.   IT SCANS THE CONDITION ITSELF, AND IT MOST
                              0000     ;        CASES (B,NB,DIF,IDN,DF,NDF ARE THE EXCEPTIONS) IT ALLOWS
                              0000     ;        THE PARSER TO EVALUATE THE ASSOCIATED EXPRESSION.
                              0000     ;
                              0000     ;--
                              0000
                              0000     IFHD1::                                           ;IF_HEAD = KIF
0000'CF    010F'CF     9E     0000          MOVAB   W^IS_TRUE,W^MAC$GL_IF_CNDPT ;PRESET IN CASE OF ERROR
               FFF6'   30     0007          BSBW    MAC$SYMSCNUP                  ;SCAN THE CONDITION CODE
              0B 50    E9     000A          BLBC    R0,10$                        ;BRANCH IF NO CONDITION FOUND
     55    0168'CF     9E     000D          MOVAB   W^IF_COND_NAMES,R5            ; Point to condition names
               FFEB'   30     0012          BSBW    MAC$SRC_LIST                  ;LOOK UP THE ONE WE SCANNED
              08 50    E8     0015          BLBS    R0,20$                        ;BRANCH IF FOUND
                              0018     10$:  $MAC_ERR ILLIFCOND                   ; Illegal IF condition
               FFE0'   31     001D          BRW     MAC$ERRORLN                   ;ISSUE MESSAGE AND RETURN
     56      05 A1     D0     0020     20$:  MOVL    SYM$L_VAL(R1),R6             ;GET THE ROUTINE ADDRESS
               FFD9'   30     0024          BSBW    MAC$SKIPSP                    ;SKIP SPACES
         2C      5A    91     0027          CMPB    R10,#^A/,/                    ;NEXT CHAR A COMMA?
                 03    12     002A          BNEQ    30$                           ;IF NEQ NO
               FFD1'   30     002C          BSBW    MAC$GETCHR                    ;YES--SKIP IT
      22 56      1F    E5     002F     30$:  BBCC    #31,R6,40$                   ;BRANCH IF NO SPECIAL SCAN
                 66    16     0033          JSB     (R6)                          ;YES--DO SPECIAL SCAN
      21 6B      16    E0     0035          BBS     #FLG$V_IIF,(R11),50$          ;IS THIS A .IIF?
0000'CF    0000'CF     D0     0039          MOVL    W^MAC$GL_LINEPT,W^MAC$GL_ERRPT ;NO--SAVE LINE POSITION
               FFBD'   30     0040          BSBW    MAC$SKIPSP                    ;SKIP SPACES
         0D      5A    91     0043          CMPB    R10,#CR                       ;WE SHOULD BE AT END OF LINE
                 12    13     0046          BEQL    50$                           ;IF EQL ALL IS WELL
                              0048          $MAC_ERR IFDIRSYNX                    ; No--If directive syntax error
               FFB0'   30     004D          BSBW    MAC$ERRORLN                   ;ISSUE MES^AGE TO PASS 2
         5A      0D    9A     0050          MOVZBL  #CR,R10                       ;FORCE NEW LINE
                 05    11     0053          BRB     50$                           ;CONTINUE
                              0055     ;
                              0055     ; NO SPECIAL SCANNING
                              0055     ;
   0000'CF       56    D0     0055     40$:  MOVL    R6,W^MAC$GL_IF_CNDPT         ;SET CONDITION TEST POINTER
  6B   00800040 8F     CA     005A     50$:  BICL2   #FLG$M_IFSTAT!FLG$M_EVALEXPR,(R11) ;NOT IN AN IF AND DO
                              0061                                                ;NOT OUTPUT EXPRESSIONS
         6B      04    C8     0061          BISL2   #FLG$M_COMPEXPR,(R11)         ;ASSUME COMPILE TIME EXPRESSION
                       05     0064          RSB
                              0065
                              0065     ;++
                              0065     ; FUNCTIONAL DESCRIPTION:
                              0065     ;
                              0065     ;        IFSYNT IS CALLED IF THERE IS A SYNTAX ERROR IN A CONDITIONAL
                              0065     ;        ASSEMBLY STATEMENT.  THE ERROR IS REPORTED, AND THE CONDITION
                              0065     ;        IS THEN PROCESSED.
                              0065     ;
                              0065     ;--
                              0065
                              0065     IFSYNT::                                          ;IF_STATE = IF_HEAD ERRO2
                              0065          $MAC_ERR IFDIRSYNX                    ; Get the message code
   FF93'  30  006A   211        BSBW    MAC$ERRORLN                   ;ISSUE MESSAGE TO PASS 2
      00  11  006D   212        BRB     IF                            ;PROCESS THE CONDITIONAL ASSEMBLY
```

```
MA
Sy

PS
RD
RD
RD
RD
RD
RD
RD
RD
RE
SC
SE
ST
SY
SY
SY
SY
SY
SY
SY
SY
SY
SY
SY
SY
SY
SY
SY
SY
SY
SY
SY
SY
SY
SY
SY
SY
SY
SY
SY
SY
SY
SY
SY
SY
SY
SY
SY
SY
SY
SY
SY
SY
SY
SY
TA
TR
X1
```

```
                               006F             .SBTTL  IF DIRECTIVE ROUTINES
                               006F
                               006F      ;++
                               006F      ; FUNCTIONAL DESCRIPTION:
                               006F      ;
                               006F      ;        THIS IS THE HEART OF THE CONDITIONAL ASSEMBLY PROCESSOR.  THIS
                               006F      ;        ROUTINE CHECKS THE RESULT OF THE IF EXPRESSION AND FALLS INTO
                               006F      ;        THE 'SCAN_FALSE_CODE'  ROUTINE WHICH SCANS THE CODE LOOKING
                               006F      ;        FOR A CHANCE TO RESUME ASSEMBLING.
                               006F      ;
                               006F      ;--
                               006F
                               006F      IF::                                               ;IF_STATE = IF_HEAD EXPR DEOL
        56   FFFC'CF47   D0   006F              MOVL    W^MAC$AL_VALSTACK-4[R7],R6 ;GET THE EXPRESSION
                               0075              $INTOUT_LW INT$_PRIL,R6          ;PRINT THE EXPRESSION VALUE
          08 6B    02    E0   007D              BBS     #FLG$V_COMPEXPR,(R11),10$ ;BRANCH IF COMPILE TIME EXPRESSION
                               0081              $MAC_ERR IFEXPRNABS              ; No--get the message code
                 FF77'   30   0086              BSBW    MAC$ERRORPT              ;ISSUE ERROR MESSAGE
             50    56    D0   0089      10$:    MOVL    R6,R0                    ;COPY THE VALUE FOR CONDITION CHECKER
                               008C
                               008C      IFSPL::                                            ;IF_STATE = IF_HEAD DEOL
            0000'DF   16      008C              JSB     @W^MAC$GL_IF_CNDPT       ;CALL THE CONDITION CHECKER
               01D9   30      0090              BSBW    IF_LIST_CND_CHK          ;CHECK  IF LISTING CONDITIONALS
            0000'CF   D4      0093              CLRL    W^MAC$GL_IF_COUNT        ;CLEAR COUNT OF CONDITIONALS WITHIN FALSE CO
         01 0000'CF   E8      0097              BLBS    W^MAC$GL_IF_VALUE,10$    ;BRANCH IF RESULT IS FALSE
                 05           009C              RSB                              ;TRUE--RETURN TO ASSEMBLE CODE
                               009D      10$:
```

```
                        009D    ;
                        009D    ; SCAN THROUGH THE  FALSE CODE, LOOKING FOR A CHANCE TO START ASSEMBLING
                        009D    ; (THE MATCHING .ENDC)
                        009D    ;
                        009D
                        009D    SCAN_FALSE_CODE:
          01CC   30     009D            BSBW    IF_LIST_CND_CHK         ;SEE ABOUT LISTING CONDITIONALS
         FF5D'   30     00A0    10$:    BSBW    MAC$SYMSCNUP            ;Check for (non-local) label
         10 50   E8     00A3            BLBS    R0,20$
         FF57'   30     00A6            BSBW    MAC$LCLSKIP             ;Try for local label
         2F 50   E9     00A9            BLBC    R0,40$
         FF51'   30     00AC            BSBW    MAC$SKIPSP
      3A    5A   91     00AF            CMPB    R10,#^A/:/              ;Ensure presence of Colon
            0A   13     00B2            BEQL    25$
            25   11     00B4            BRB     40$
         FF47'   30     00B6    20$:    BSBW    MAC$SKIPSP              ;Skip any spaces
      3A    5A   91     00B9            CMPB    R10,#^A/:/              ;Presence of Colon indicates label
            05   12     00BC            BNEQ    30$
         FF3F'   30     00BE    25$:    BSBW    MAC$GETCHR              ;Found a label -- go back for more
            DD   11     00C1            BRB     10$
  55   01FA'CF   9E     00C3    30$:    MOVAB   W^IF_SPL_KEYWORDS,R5    ;We have a symbol -- look it up
         FF35'   30     00C8            BSBW    MAC$SRC_LIST            ;...
         0D 50   E9     00CB            BLBC    R0,40$                  ;BRANCH IF NOT FOUND
         05 A1   DD     00CE            PUSHL   SYM$L_VAL(R1)           ;FOUND--STACK ROUTINE ADDRESS
         FF2C'   30     00D1            BSBW    MAC$CREF_DIR            ;CROSS-REF IT IF CREFFING DIRECTIVES
                        00D4                                            ;(R1 POINTS TO SYMBOL BLOCK)
                        00D4            $INTOUT_X INT$_CHKL             ;PRINT SOURCE LINES NOT ASSEMBLED
                        00DA    ;
                        00DA    ; BRANCH TO THE ROUTINE FOR THE SPECIAL SYMBOL.  THE ROUTINE WILL EITHER
                        00DA    ; BRANCH BACK TO SCAN_FALSE_CODE TO CONTINUE LOOKING FOR TRUTHE, OR
                        00DA    ; IT WILL RETURN IF IT IS TIME TO ASSEMBLE CODE AGAIN.
                        00DA    ;
            05          00DA            RSB                             ;GO TO THE SPECIAL ROUTINE
       0000'CF   DD     00DB    40$:    PUSHL   W^MAC$GL_INPUTP         ;STACK INPUT BLOCK POINTER
      5A    0D   9A     00DF            MOVZBL  #CR,R10                 ;FORCE NEW LINE
         FF1B'   30     00E2            BSBW    MAC$GETCHR              ;READ IT
  8E   0000'CF   D1     00E5            CMPL    W^MAC$GL_INPUTP,(SP)+   ;WAS THERE A CONTEXT CHANGE?
            B1   13     00EA            BEQL    SCAN_FALSE_CODE         ;IF EQL NO--KEEP SCANNING
            05          00EC            RSB                             ;YES--RETURN
```

```
             00ED           .SBTTL  "IF" CONDITION ROUTINES--EQ,NE,GT,LE,GE,LT
             00ED
             00ED  ;++
             00ED  ; FUNCTIONAL DESCRIPTION:
             00ED  ;
             00ED  ;        THESE ROUTINES TEST THE EXPRESSION CONTAINED IN R0 FOR THE
             00ED  ;        CONDITION DESIRED. THE LOW BIT OF 'MAC$GL_IF_VALUE' WILL
             00ED  ;        BE CLEARED IF IT TESTS TRUE, AND SET IF IT TESTS FALSE.
             00ED  ;
             00ED  ;--
             00ED
             00ED  IF_EQUAL:
  50    D5   00ED          TSTL    R0                      ;CHECK CONDITION
  1E    13   00EF          BEQL    IS_TRUE                 ;IF EQL IS TRUE
  20    11   00F1          BRB     IS_FALSE                ;ELSE IS FALSE
             00F3
             00F3  IF_NOT_EQUAL:
  50    D5   00F3          TSTL    R0                      ;CHECK CONDITION
  18    12   00F5          BNEQ    IS_TRUE                 ;IF NEQ IS TRUE
  1A    11   00F7          BRB     IS_FALSE                ;ELSE IS FALSE
             00F9
             00F9  IF_GREATER:
  50    D5   00F9          TSTL    R0                      ;CHECK CONDITION
  12    14   00FB          BGTR    IS_TRUE                 ;IF GTR IS TRUE
  14    11   00FD          BRB     IS_FALSE
```

```
                       00FF   IF_LESS_EQUAL:
            50    D5   00FF           TSTL    R0                          ;CHECK CONDITION
            0C    15   0101           BLEQ    IS_TRUE                     ;IF LEQ IS TRUE
            0E    11   0103           BRB     IS_FALSE                    ;ELSE IS FALSE
                       0105
                       0105   IF_LESS_THAN:
            50    D5   0105           TSTL    R0                          ;CHECK CONDITION
            06    19   0107           BLSS    IS_TRUE                     ;IF LSS IS TRUE
            08    11   0109           BRB     IS_FALSE                    ;ELSE IS FALSE
                       010B
                       010B   IF_GTR_EQUAL:
            50    D5   010B           TSTL    R0                          ;CHECK CONDITION
            04    19   010D           BLSS    IS_FALSE                    ;IF LSS THEN FALSE
                       010F   ;**;    BRB     IS_TRUE                     ;ELSE IS TRUE
                       010F
                       010F   IS_TRUE:
            50    D4   010F           CLRL    R0                          ;SET FOR TRUTH
            03    11   0111           BRB     TRUE_FALSE
                       0113
                       0113   IS_FALSE:
      50    01    9A   0113           MOVZBL  #1,R0                       ;SET FOR FALSE
                       0116   TRUE_FALSE:
51  0000'CF  01   9C   0116           ROTL    #1,W^MAC$GL_IF_VALUE,R1 ;MAKE ROOM FOR NEW RESULT
0000'CF  51  50   C9   011C           BISL3   R0,R1,W^MAC$GL_IF_VALUE ;OR IN NEW CONDITION AND STORE IT
       0000'CF    D6   0122           INCL    W^MAC$GL_IF_LEVEL       ;COUNT NEW NESTING LEVEL
   20  0000'CF    D1   0126           CMPL    W^MAC$GL_IF_LEVEL,#32   ;NESTING EXCEEDED?
            08    15   012B           BLEQ    10$                     ;IF LEQ NO
                       012D           $MAC_ERR IFLEVLXCED             ; Yes--get message code
         FECB'    31   0132           BRW     MAC$ERRORLN             ;ISSUE MESSAGE TO PASS 2 AND RETURN
                  05   0135   10$:    RSB
```

```
                         0136          .SBTTL  'IF' CONDITION ROUTINES--IF_DEFINED
                         0136
                         0136    ;++
                         0136    ; FUNCTIONAL DESCRIPTION:
                         0136    ;
                         0136    ;         THIS ROUTINE SETS THE POINTER MAC$GL_IF_CNDPT TO POINT
                         0136    ;         TO IS_TRUE OR IS_FALSE, DEPENDING ON WHETHER THE SYMBOL
                         0136    ;         IS DEFINED OR NOT.
                         0136    ;
                         0136    ;--
                         0136
                         0136    IF_DEFINED:
        FFD5 CF     9F   0136            PUSHAB  W^IS_TRUE                 ;IF DEFINED
        FFD5 CF     9F   013A            PUSHAB  W^IS_FALSE                ;IF NOT DEFINED
             08     11   013E            BRB     IF_DF
                         0140
                         0140    IF_NOT_DEFINED:
        FFCF CF     9F   0140            PUSHAB  W^IS_FALSE                ;IF DEFINED
        FFC7 CF     9F   0144            PUSHAB  W^IS_TRUE                 ;IF NOT DEFINED
          FEB5'     30   0148    IF_DF:  BSBW    MAC$SYMSCNUP              ;SCAN A SYMBOL
          08 50     E8   014B            BLBS    R0,10$                    ;BRANCH IF WE SCANNED ONE
                         014E            $MAC_ERR ILLIFCOND                ; No--get message code
        8E   8E     D1   0153            CMPL    (SP)+,(SP)+               ;CLEAR ROUTINE ADDRESSES
          FEA7'     31   0156            BRW     MAC$ERRORLN               ;ISSUE TO PASS 2 AND RETURN
          FEA4'     30   0159    10$:    BSBW    MAC$SRCUSRSYMTB           ;SEARCH SYMBOL TABLE FOR IT
          05 50     E9   015C            BLBC    R0,20$                    ;BRANCH IF NOT FOUND
08 09 A1     00     E0   015F            BBS     #SYM$V_DEF,SYM$W_FLAG(R1),30$ ;BRANCH IF SYMBOL IS DEFINED
       0000'CF    8ED0   0164    20$:    POPL    W^MAC$GL_IF_CNDPT         ;NOT DEFINED--GET RESULT
             8E     D5   0169            TSTL    (SP)+                     ;CLEAR OTHER RESULT
                    05   016B            RSB
             8E     D5   016C    30$:    TSTL    (SP)+                     ;CLEAR NOT DEFINED RESULT
       0000'CF    8ED0   016E            POPL    W^MAC$GL_IF_CNDPT         ;GET DEFINED RESULT
                    05   0173            RSB
```

```
                 0174              .SBTTL  'IF' CONDITION ROUTINES--IF_BLANK
                 0174
                 0174      ;++
                 0174      ; FUNCTIONAL DESCRIPTION:
                 0174      ;
                 0174      ;       THIS ROUTINE SETS THE POINTER MAC$GL_IF_CNDPT TO POINT
                 0174      ;       TO IS_TRUE OR IS_FALSE, DEPENDING ON WHETHER OR NOT THE
                 0174      ;       ARGUMENT IS BLANK OR NOT.
                 0174      ;
                 0174      ;--
                 0174
                 0174      IF_BLANK:
  FF97 CF    9F  0174              PUSHAB  W^IS_TRUE                       ;IF BLANK
  FF97 CF    9F  0178              PUSHAB  W^IS_FALSE                      ;IF NOT BLANK
       08    11  017C              BRB     IF_B                            ;JOIN COMMON CODE
                 017E
                 017E      IF_NOT_BLANK:
  FF91 CF    9F  017E              PUSHAB  W^IS_FALSE                      ;IF BLANK
  FF89 CF    9F  0182              PUSHAB  W^IS_TRUE                       ;IF NOT BLANK
00 6B    17  E3  0186      IF_B:   BBCS    #FLG$V_IFSTAT,(R11),.+1 ;FLAG WE ARE IN AN IF
    FE73'    30  018A              BSBW    MAC$MAC_ARG_SCN                 ;SCAN THE ARGUMENT
00 6B    17  E5  018D              BBCC    #FLG$V_IFSTAT,(R11),.+1 ;NOT IN AN IF ANY MORE
       50    D5  0191              TSTL    R0                              ;WAS THE ARGUMENT BLANK?
       08    12  0193              BNEQ    10$                             ;IF NEQ NO
       8E    D5  0195              TSTL    (SP)+                           ;YES--CLEAR FALSE CONDITION
  0000'CF  8ED0  0197              POPL    W^MAC$GL_IF_CNDPT               ;SET TRUE CONDITION
             05  019C              RSB
  0000'CF  8ED0  019D      10$:    POPL    W^MAC$GL_IF_CNDPT               ;SET FALSE CONDITION
       8E    D5  01A2              TSTL    (SP)+                           ;CLEAR TRUE CONDITION
             05  01A4              RSB
```

```
                                 01A5          .SBTTL  DIRECTIVE ROUTINES--IF_IDENTICAL
                                 01A5
                                 01A5  ;++
                                 01A5  ; FUNCTIONAL DESCRIPTION:
                                 01A5  ;
                                 01A5  ;       THIS ROUTINE DETERMINES WHETHER TWO STRINGS ARE IDENTICAL
                                 01A5  ;       OR NOT, AND SETS THE APPROPRIATE ROUTINE ADDRESS INTO
                                 01A5  ;       MAC$GL_IF_CNDPT.
                                 01A5  ;
                                 01A5  ;--
                                 01A5
                                 01A5  IF_IDENTICAL:
                         5C   DD 01A5          PUSHL   R12                           ;SAVE R12
                    FF64 CF   9F 01A7          PUSHAB  W^IS_TRUE                     ;TRUE RESULT
                    FF64 CF   9F 01AB          PUSHAB  W^IS_FALSE                    ;FALSE RESULT
                         0A   11 01AF          BRB     IF_IDN                        ;GO PROCESS IT
                                 01B1
                                 01B1  IF_DIFFERENT:
                         5C   DD 01B1          PUSHL   R12                           ;SAVE R12
                    FF5C CF   9F 01B3          PUSHAB  W^IS_FALSE                    ;TRUE RESULT
                    FF54 CF   9F 01B7          PUSHAB  W^IS_TRUE                     ;FALSE RESULT
                         5C   D4 01BB  IF_IDN: CLRL    R12                           ;ASSUME NULL FIRST ARGUMENT
                  00 6B  26   E2 01BD          BBSS    #FLG$V_UPMARG,(R11),.+1 ; Get arguments upper cased
                      FE3C'   30 01C1          BSBW    MAC$MAC_ARG_SCN               ;SCAN THE FIRST ARGUMENT
                         50   DD 01C4          PUSHL   R0                            ;STACK THE LENGTH OF THE ARG
                         18   13 01C6          BEQL    20$                           ;BRANCH IF NULL ARG
                  51  50 08   C1 01C8          ADDL3   #MXB$K_BLKSIZ,R0,R1           ; Include header size
                      FE31'   30 01CC          BSBW    MAC$ALL[_BLOCK                ; Allocate memory block
                   04 A0 51   D0 01CF          MOVL    R1,MXB$[_PAGES(R0)            ; Save block size in block
                  56  50 08   C1 01D3          ADDL3   #MXB$K_B[KSIZ,R0,R6           ; Set pointer to free bytes
                      5C 56   D0 01D7          MOVL    R6,R12                        ; Save pointer
          66  0000'CF 6E      28 01DA          MOVC3   (SP),W^MAC$AB_TMPBUF,(R6) ;COPY ARG TO VIRT. MEMORY
                  00 6B  17   E3 01E0  20$:    BBCS    #FLG$V_IFSTAT,(R11),.+1 ;FLAG WITHIN AN IF
                      FE19'   30 01E4          BSBW    MAC$MAC_ARG_SCN               ;SCAN SECOND ARGUMENT
                  00 6B  17   E5 01E7          BBCC    #FLG$V_IFSTAT,(R11),.+1 ;NO LONGER WITHIN AN IF
                  00 6B  26   E5 01EB          BBCC    #FLG$V_UPMARG,(R11),.+1 ; Return normal argument processing
                      56 8ED0    01EF          POPL    R6                            ;GET LENGTH OF FIRST STRING
                     56  50   D1 01F2  50$:    CMPL    R0,R6                         ;STRINGS THE SAME LENGTH?
                         17   12 01F5          BNEQ    70$                           ;IF NEQ NO
                         50   D5 01F7          TSTL    R0                            ;YES--ARE THEY BOTH NULL?
                         0A   13 01F9          BEQL    60$                           ;IF EQL YES--THEY ARE THE SAME
  0000'CF    50    00    6C   56 2D 01FB       CMPC5   R6,(R12),#0,R0,W^MAC$AB_TMPBUF ;NO--STRINGS IDENTICAL?
                         09   12 0203          BNEQ    70$                           ;IF NEQ NO
                         8E   D5 0205  60$:    TSTL    (SP)+                         ;CLEAR FALSE RESULT
                 0000'CF 8ED0    0207          POPL    W^MAC$GL_IF_CNDPT             ;SET TRUE RESULT
                         07   11 020C          BRB     80$                           ;FINISH UP
                 0000'CF 8ED0    020E  70$:    POPL    W^MAC$GL_IF_CNDPT             ;STORE FALSE RESULT
                         8E   D5 0213          TSTL    (SP)+                         ;POP FALSE RESULT
                     50  5C   D0 0215  80$:    MOVL    R12,R0                        ;GET ADDRESS OF PAGE FOR ARG 1
                         06   13 0218          BEQL    90$                           ;IF EQL NO PAGE ALLOCATED
                     50  08   C2 021A          SUBL2   #MXB$K_BLKSIZ,R0              ; Point to base of block
                      FDE0'   30 021D          BSBW    MAC$DEAL_BLOCK                ; and deallocate
                      5C 8ED0    0220  90$:    POPL    R12                           ;RESTORE R12
                         05      0223          RSB                                   ;DONE
```

```
                              0224          .SBTTL  DIRECTIVE ROUTINES--IFF,IFT,IFTF, ENDC
                              0224
                              0224   ;++
                              0224   ; FUNCTIONAL DESCRIPTION:
                              0224   ;
                              0224   ;       THIS ROUTINE CAN BE CALLED FROM TWO PLACES: 1) THE SCAN_FALSE_CODE
                              0224   ;       ROUTINE, WHEN IT DETECTS A .IFF WHILE SCANNING FALSE CODE, OR
                              0224   ;       2) FROM THE PARSER.  IT CHECKS THE IF STATUS, AND IF WE ARE
                              0224   ;       SCANNING FALSE CODE, IT BRANCHES TO SCAN_FALSE_CODE TO CONTINUE
                              0224   ;       SCANNING FALSE CODE.  IF IT TESTS TRUE, WE RETURN TO THE PARSER
                              0224   ;       TO ASSEMBLE CODE.
                              0224   ;
                              0224   ;--
                              0224
                              0224   IFF::                                          ;DIRECTIVE = KIFF
                  5A     10   0224          BSBB    CHECK_IF_STATUS                ;CHECK 'IF' STATUS
          41 0000'CF     E8   0226          BLBS    W^MAC$GL_IF_VALUE,IF_LIST_CND_CHK ;BRANCH IF NOT IN FALSE CODE
                              022B   GO_SCAN_FALSE:
                FE6F     31   022B          BRW     SCAN_FALSE_CODE                ;ELSE CONTINUE SCANNING FALSE CODE
                              022E
                              022E   IFT::                                          ;DIRECTIVE = KIFT
                  50     10   022E          BSBB    CHECK_IF_STATUS                ;CHECK 'IF' STATUS
          F6 0000'CF     E8   0230          BLBS    W^MAC$GL_IF_VALUE,GO_SCAN_FALSE ;BRANCH IF WITHIN FALSE
                  35     11   0235          BRB     IF_LIST_CND_CHK                ;ELSE RETURN TO ASSEMBLE CODE
                              0237
                              0237   IFTF::                                         ;DIRECTIVE = KIFTF
                  47     10   0237          BSBB    CHECK_IF_STATUS                ;CHECK 'IF' STATUS
                  31     11   0239          BRB     IF_LIST_CND_CHK                ;CHECK LISTING AND RETURN
                              023B
                              023B   ENDC::                                         ;DIRECTIVE = KENDC
    56    0000'CF    01  C3   023B          SUBL3   #1,W^MAC$GL_IF_LEVEL,R6 ;DECREMENT IF LEVEL AND CHECK
                     08  18   0241          BGEQ    10$                            ;IF GEQ WITHIN AN IF
                              0243          $MAC_ERR NOTINANIF                     ; No--get message code
               FDB5'     31   0248          BRW     MAC$ERRORLN                    ;ISSUE MESSAGE TO PASS 2 AND RETURN
    55    0000'CF    01  C3   024B   10$:   SUBL3   #1,W^MAC$GL_IF_COUNT,R5 ;SEE IF IN NESTED FALSE CONDITIONAL
                     07  19   0251          BLSS    20$                            ;IF LSS NO
          0000'CF    55  D0   0253          MOVL    R5,W^MAC$GL_IF_COUNT           ;YES--UPDATE NESTING COUNT
                     D1  11   0258          BRB     GO_SCAN_FALSE                  ;AND CONTINUE SCANNING FALSE CODE
          0000'CF    56  D0   025A   20$:   MOVL    R6,W^MAC$GL_IF_LEVEL           ;UPDATE IF LEVEL
    50    0000'CF    01  CB   025F          BICL3   #1,W^MAC$GL_IF_VALUE,R0 ;PREPARE TO BRING TRUTH INTO HIGH BIT
 0000'CF     50   FF 8F  9C   0265          ROTL    #-1,R0,W^MAC$GL_IF_VALUE ;DO IT NOW
                              026C   ;**;   BRB     IF_LIST_CND_CHK                ;CHECK LISTING STATUS AND RETURN
                              026C
                              026C   ;++
                              026C   ; FUNCTIONAL DESCRIPTION:
                              026C   ;
                              026C   ;       IF NOT LISTING CONDITIONALS, CODE IS EMITTED TO PASS 2 TO
                              026C   ;       CLEAR THE LISTING FLAG, MAC$GL_LIST_IT.
                              026C   ;
                              026C   ;--
                              026C
                              026C   IF_LIST_CND_CHK:
          0E 0005'CF     E8   026C          BLBS    W^LST$G_CONDITION+SYM$L_VAL,CK_EXIT ;BRANCH IF LISTING
                              0271          $INTOUT_LW INT$_SETLONG,<#0,#MAC$GL_LIST_IT> ;NO--
                         05   027F   CK_EXIT:RSB
                              0280
                              0280
```

```
                    0280  ;++
                    0280  ; FUNCTIONAL DESCRIPTION:
                    0280  ;
                    0280  ;       THIS ROUTINE CHECKS TO ENSURE WE ARE IN AN IF STATEMENT.
                    0280  ;       IF WE ARE NOT, IT ISSUES AN ERROR MESSAGE TO PASS 2
                    0280  ;       AND RETURNS.  IF WE ARE, THEN IF WE ARE SKIPPING CODE, THE
                    0280  ;       STACK IS POPPED AND WE BRANCH TO SCAN_FALSE_CODE TO CONTINUE
                    0280  ;       SKIPPING CODE.
                    0280  ;
                    0280  ;--
                    0280
                    0280  CHECK_IF_STATUS:
 0000'CF    D5      0280          TSTL    W^MAC$GL_IF_LEVEL       ;ARE WE IN AN IF?
      08    14      0284          BGTR    10$                     ;IF GTR YES
                    0286          $MAC_ERR NOTINANIF              ; No--get message code
   FD72'    31      028B          BRW     MAC$ERRORLN             ;ISSUE MESSAGE AND RETURN
 0000'CF    D5      028E  10$:    TSTL    W^MAC$GL_IF_COUNT       ;INSIDE NESTED FALSE CONDITIONAL?
      05    15      0292          BLEQ    20$                     ;IF LEQ NO
      8E    D5      0294          TSTL    (SP)+                   ;YES--CLEAR RETURN
    FE04    31      0296          BRW     SCAN_FALSE_CODE         ;AND CONTINUE SCANNING FALSE CODE
            05      0299  20$:    RSB
                    029A
                    029A  ;++
                    029A  ; FUNCTIONAL DESCRIPTION:
                    029A  ;
                    029A  ;       THIS ROUTINE IS CALLED IF A .END STATMENT IS ENCOUNTERED
                    029A  ;       WHILE SCANNING THE FALSE CONDITIONAL CODE.
                    029A  ;
                    029A  ;--
                    029A
                    029A  IF_ERROR:
 0000'CF    D4      029A          CLRL    W^MAC$GL_IF_VALUE       ;EVERYTHING IS TRUE
                    029E          $MAC_ERR UNTERMCOND             ; Get message code
   FD5A'    31      02A3          BRW     MAC$ERRORLN             ;ISSUE MESSAGE AND RETURN
                    02A6
                    02A6  ;++
                    02A6  ; FUNCTIONAL DESCRIPTION:
                    02A6  ;
                    02A6  ;       THIS ROUTINE IS CALLED IF A .IF STATEMENT IS ENCOUNTERED
                    02A6  ;       WHILE SCANNING THE FALSE CONDITION CODE.
                    02A6  ;
                    02A6  ;--
                    02A6
                    02A6  IF_IN_AN_IF:
 0000'CF    D6      02A6          INCL    W^MAC$GL_IF_COUNT       ;BUMP FALSE CONDITIONAL NESTING COUNT
    FDF0    31      02AA          BRW     SCAN_FALSE_CODE         ;CONTINUE SCANNING FALSE CODE
```

```
                                02AD              .SBTTL  .IIF DIRECTIVE ROUTINES
                                02AD
                                02AD      ;++
                                02AD      ; FUNCTIONAL DESCRIPTION:
                                02AD      ;
                                02AD      ;       IIF IS CALLED WHEN A .IIF DIRECTIVE IS DETECTED.  THE IIF_HEAD
                                02AD      ;       IS SCANNED.  THE PARSER WILL THEN CALL IIF1 TO FINISH PROCESSING
                                02AD      ;       THE .IIF DIRECTIVE.
                                02AD      ;
                                02AD      ;--
                                02AD
                                02AD      IIF::                                           ;IIF_HEAD = KIIF
               00 6B    16   E3 02AD              BBCS    #FLG$V_IIF,(R11),.+1            ;FLAG THIS IS .IIF
                      FD4C   30 02B1              BSBW    IFHD1                           ;SCAN THE CONDITION
               00 6B    16   E5 02B4              BBCC    #FLG$V_IIF,(R11),.+1            ;CLEAR .IIF FLAG
                      FFB1   31 02B8              BRW     IF_LIST_CND_CHK                 ;CHECK LISTING AND RETURN
                                02BB
                                02BB      IIF1::                                          ;IIF_STAT = IIF_HEAD EXPR DCOMMA
               08 6B    02   E0 02BB              BBS     #FLG$V_COMPEXPR,(R11),10$ ;BRANCH IF COMPILE TIME EXPRESSION
                                02BF              $MAC_ERR IFEXPRNABS                     ; No--get message code
                     FD39'   30 02C4              BSBW    MAC$ERRORLN                     ;ISSUE TO PASS 2
            50    FFFC'CF47  D0 02C7      10$:    MOVL    W^MAC$AL_VALSTACK-4[R7],R0 ;GET THE VALUE
                   0000'DF   16 02CD              JSB     @W^MAC$GL_IF_CNDPT              ;CALL THE ROUTINE TO EVALUATE CONDITION
              50   0000'CF   D0 02D1              MOVL    W^MAC$GL_IF_VALUE,R0            ;GET THE 'IF' VALUE
              51    50    01 CB 02D6              BICL3   #1,R0,R1                        ;SET TO BRING TRUTH INTO HI BIT
      0000'CF  51    FF 8F   9C 02DA              ROTL    #-1,R1,W^MAC$GL_IF_VALUE ;DO IT AND STORE
                   0000'CF   D7 02E1              DECL    W^MAC$GL_IF_LEVEL               ;DROP DOWN AN IF LEVEL
                     0D 50   E8 02E5              BLBS    R0,IIF_FALSE                    ;BRANCH IF FALSE
                        0E   11 02E8              BRB     IIF_TRUE                        ;GO TO TRUE EXIT
                                02EA
                                02EA      IIF2::                                          ;IIF_STAT = IIF_HEAD DCOMMA
     00000113'8F   0000'CF   D1 02EA              CMPL    W^MAC$GL_IF_CNDPT,#IS_FALSE ; WAS CONDITION FALSE?
                        03   12 02F3              BNEQ    IIF_TRUE                        ;BRANCH IF NOT
                                02F5      IIF_FALSE:
                  5A    0D   9A 02F5              MOVZBL  #CR,R10                         ;FORCE NEW LINE
                                02F8      IIF_TRUE:
               00 6B    01   E3 02F8              BBCS    #FLG$V_BOL,(R11),.+1            ;SET BOL FLAG
               00 6B    0D   E5 02FC              BBCC    #FLG$V_OPRND,(R11),.+1   ;NOT IN OPERAND FIELD
               00 6B    06   E3 0300              BBCS    #FLG$V_EVALEXPR,(R11),.+1 ;ALLOW EXPRESSION EVALUATION AGAIN.
                             05 0304              RSB
                                0305
                                0305              .END
```

```
$COUNT          = 0000003B              FLG$M_NOREF     = 01000000      FLG$V_OPTVFLIDX = 0000002C
ARG$K_SIZE      = 000003E8              FLG$M_NTYPEPC   = 00000020      FLG$V_ORDLST    = 00000011
AUD$K_SIZE      = 00000010              FLG$M_NULCHR    = 00040000      FLG$V_P2        = 0000000E
BLNK            = 00000020              FLG$M_OBJXST    = 00200000      FLG$V_RPTIRP    = 0000001C
CHECK_IF_STATUS   00000280 R      04    FLG$M_OPNDCHK   = 00000100      FLG$V_SEQFIL    = 00000019
CHR$M_COMMA_CR  = 00000020              FLG$M_OPRND     = 00002000      FLG$V_SKAN      = 0000000F
CHR$M_ILL_CHR   = 00000040              FLG$M_OPTVFLIDX = 00001000      FLG$V_SPECOP    = 00000022
CHR$M_NUM_BER   = 00000010              FLG$M_ORDLST    = 00020000      FLG$V_SPLALL    = 0000001A
CHR$M_SPA_MSK   = 00000001              FLG$M_P2        = 00004000      FLG$V_STOIMF    = 00000012
CHR$M_SYM_CH1   = 00000008              FLG$M_RPTIRP    = 10000000      FLG$V_SYM2COL   = 0000002A
CHR$M_SYM_CHR   = 00000004              FLG$M_SEQFIL    = 02000000      FLG$V_TOCFLG    = 00000013
CHR$M_SYM_DLM   = 00000002              FLG$M_SKAN      = 00008000      FLG$V_UPAFLG    = 00000024
CHR$V_COMMA_CR  = 00000005              FLG$M_SPECOP    = 00000004      FLG$V_UPDFIL    = 00000027
CHR$V_CVTLWC    = 00000061              FLG$M_SPLALL    = 04000000      FLG$V_UPMARG    = 00000026
CHR$V_ILL_CHR   = 00000006              FLG$M_STOIMF    = 00040000      FLG$V_XCRF      = 0000001F
CHR$V_NOCVT     = 0000007F              FLG$M_SYM2COL   = 00000400      GO_SCAN_FALSE     0000022B R      04
CHR$V_NUM_BER   = 00000004              FLG$M_TOCFLG    = 00080000      HASHSZ          = 0000007F
CHR$V_SPA_MSK   = 00000000              FLG$M_UPAFLG    = 00000010      HYPHEN          = 0000002D
CHR$V_SYM_CH1   = 00000003              FLG$M_UPDFIL    = 00000080      IF                0000006F RG     04
CHR$V_SYM_CHR   = 00000002              FLG$M_UPMARG    = 00000040      IFF               00000224 RG     04
CHR$V_SYM_DLM   = 00000001              FLG$M_XCRF      = 80000000      IFHD1             00000000 RG     04
CK_EXIT           0000027F R      04    FLG$V_ALLCHR    = 00000000      IFSPL             0000008C RG     04
CNT             = 00000002              FLG$V_BOL       = 00000001      IFSYNT            00000065 RG     04
CR              = 0000000D              FLG$V_CHKLPND   = 00000014      IFT               0000022E RG     04
ENDC              0000023B RG     04    FLG$V_COMPEXPR  = 00000002      IFTF              00000237 RG     04
ERR             = 00000000              FLG$V_CONT      = 00000003      IF_B              00000186 R      04
FF              = 0000000C              FLG$V_CRF       = 0000001E      IF_BLANK          00000174 R      04
FLG$M_ALLCHR    = 00000001              FLG$V_CRSEEN    = 00000020      IF_COND_NAMES     00000168 RG     03
FLG$M_BOL       = 00000002              FLG$V_DATRPT    = 00000004      IF_DEFINED        00000136 R      04
FLG$M_CHKLPND   = 00100000              FLG$V_DBGOUT    = 0000002E      IF_DF             00000148 R      04
FLG$M_COMPEXPR  = 00000004              FLG$V_DLIMSTR   = 0000002F      IF_DIFFERENT      000001B1 R      04
FLG$M_CONT      = 00000008              FLG$V_ENDMCH    = 00000005      IF_EQUAL          000000ED R      04
FLG$M_CRF       = 40000000              FLG$V_EVALEXPR  = 00000006      IF_ERROR          0000029A R      04
FLG$M_CRSEEN    = 00000001              FLG$V_EXPOPT    = 00000007      IF_GREATER        000000F9 R      04
FLG$M_DATRPT    = 00000010              FLG$V_EXTERR    = 00000030      IF_GTR_EQUAL      0000010B R      04
FLG$M_DBGOUT    = 00004000              FLG$V_EXTWRN    = 00000031      IF_IDENTICAL      000001A5 R      04
FLG$M_DLIMSTR   = 00008000              FLG$V_FIRSTLN   = 00000029      IF_IDN            000001BB R      04
FLG$M_ENDMCH    = 00000020              FLG$V_IFSTAT    = 00000017      IF_IN_AN_IF       000002A6 R      04
FLG$M_EVALEXPR  = 00000040              FLG$V_IIF       = 00000016      IF_LESS_EQUAL     000000FF R      04
FLG$M_EXPOPT    = 00000080              FLG$V_INSERT    = 00000008      IF_LESS_THAN      00000105 R      04
FLG$M_EXTERR    = 00010000              FLG$V_IRPC      = 0000001D      IF_LIST_CND_CHK   0000026C R      04
FLG$M_EXTWRN    = 00020000              FLG$V_LEXOP     = 00000021      IF_NOT_BLANK      0000017E R      04
FLG$M_FIRSTLN   = 00000200              FLG$V_LSTXST    = 00000009      IF_NOT_DEFINED    00000140 R      04
FLG$M_IFSTAT    = 00800000              FLG$V_MAC2COL   = 0000002B      IF_NOT_EQUAL      000000F3 R      04
FLG$M_IIF       = 00400000              FLG$V_MACL      = 0000000B      IF_SPECIAL      = 80000000
FLG$M_INSERT    = 00000100              FLG$V_MACLTB    = 0000001B      IF_SPL_KEYWORDS   000001FA RG     03
FLG$M_IRPC      = 20000000              FLG$V_MACTXT    = 00000010      IIF               000002AD RG     04
FLG$M_LEXOP     = 00000002              FLG$V_MEBLST    = 0000000C      IIF1              000002BB RG     04
FLG$M_LSTXST    = 00000200              FLG$V_MOREARG   = 0000002D      IIF2              000002EA RG     04
FLG$M_MAC2COL   = 00000800              FLG$V_MOREINP   = 00000023      IIF_FALSE         000002F5 R      04
FLG$M_MACL      = 00000800              FLG$V_NEWPND    = 0000000A      IIF_TRUE          000002F8 R      04
FLG$M_MACLTB    = 08000000              FLG$V_NOREF     = 00000018      INP$K_BUFSIZ    = 000003E8
FLG$M_MACTXT    = 00010000              FLG$V_NTYPEPC   = 00000025      INSYMC          = 00000005
FLG$M_MEBLST    = 00001000              FLG$V_NULCHR    = 00000032      INSYMP          = 000001FA R      03
FLG$M_MOREARG   = 00002000              FLG$V_OBJXST    = 00000015      INSYTM          = 000001FA R      03
FLG$M_MOREINP   = 00000008              FLG$V_OPNDCHK   = 00000028      INT$K_BUFSIZ    = 000013F4
FLG$M_NEWPND    = 00000400              FLG$V_OPRND     = 0000000D      INT$K_BUFWRN    = 00001390
```

```
                                                                 B 16
MAC$ACTIF                         CONDITIONAL STATEMENT PROCESSOR          16-SEP-1984 01:59:08  VAX/VMS Macro V04-00       Page  18
Symbol table                                                                5-SEP-1984 01:46:51  [MACRO.SRC]ACTIF.MAR;1            (12)

INT$_ADD        = 00000001        INT$_WRN        = 00000038              MXB$L_LINK         00000000
INT$_AND        = 00000002        INT$_XOR        = 0000000B              MXB$L_PAGES        00000004
INT$_ASH        = 00000003        IS_FALSE          00000113 R      04    OBJ$K_BUFSIZ     = 00000200
INT$_ASN        = 0000000C        IS_TRUE           0000010F R      04    OPF$M_LASTOPR    = 00002000
INT$_AUGPC      = 0000000D        LST$G_CONDITION   ********   X    04    OPF$M_OPTEXP     = 00001000
INT$_BDST       = 0000000E        LST$K_BUFSIZ    = 00000086              OPF$V_LASTOPR    = 0000000D
INT$_CHKL       = 0000000F        LST$K_L_P_PAGE  = 0000003C              OPF$V_OPTEXP     = 0000000C
INT$_DIV        = 00000004        LST$K_TITLE_SIZ=  00000028              PSC$B_NAME         00000004
INT$_END        = 00000010        MAB$B_ARGNO       00000005              PSC$B_SEG          0000000C
INT$_EPT        = 00000011        MAB$B_NAME        00000004              PSC$B_UNUSED       0000000B
INT$_ERR        = 00000012        MAB$K_BLKSIZ      0000000C              PSC$K_BLKSIZ       00000013
INT$_ETX        = 00000013        MAB$L_DVPTR       00000006              PSC$K_NO_OPTNS   = 0000000A
INT$_FNEWL      = 00000014        MAB$L_LINK        00000000              PSC$L_CURLOC       0000000F
INT$_ILG        = 00000000        MAB$W_DVLEN       00000006              PSC$L_LINK         00000000
INT$_INFO       = 0000003A        MAC$AB_TMPBUF     ********   X    04    PSC$L_MAXLGTH      00000005
INT$_LGLAB      = 00000015        MAC$AL_BLOCK      ********   X    04    PSC$M_ABS        = FFFFFFF7
INT$_MACL       = 00000016        MAC$AL_VALSTACK   ********   X    04    PSC$M_ALIGNFLG   = 00004000
INT$_MUL        = 00000005        MAC$CREF_DIR      ********   X    04    PSC$M_ALLOPTNS   = 000003FF
INT$_NEG        = 00000006        MAC$DEAL_BLOCK    ********   X    04    PSC$M_BYTE       = 00004000
INT$_NEWL       = 00000017        MAC$ERRORLN       ********   X    04    PSC$M_CON        = FFFFFFFB
INT$_NEWP       = 00000018        MAC$ERRORPT       ********   X    04    PSC$M_DEFAULT    = 000001C8
INT$_NOT        = 00000007        MAC$GETCHR        ********   X    04    PSC$M_EXE        = 000000C0
INT$_OP         = 00000019        MAC$GL_ERRPT      ********   X    04    PSC$M_GBL        = 00000010
INT$_OR         = 00000008        MAC$GL_IF_CNDPT   ********   X    04    PSC$M_LCL        = FFFFFFEF
INT$_PRIL       = 0000001A        MAC$GL_IF_COUNT   ********   X    04    PSC$M_LIB        = 00000002
INT$_PRT        = 0000001B        MAC$GL_IF_LEVEL   ********   X    04    PSC$M_LONG       = 00004800
INT$_PSECT      = 0000001C        MAC$GL_IF_VALUE   ********   X    04    PSC$M_NOEXE      = FFFFFFBF
INT$_REDEF      = 0000001D        MAC$GL_INPUTP     ********   X    04    PSC$M_NOPIC      = FFFFFFFE
INT$_REF        = 0000001E        MAC$GL_LINEPT     ********   X    04    PSC$M_NORD       = FFFFFF7F
INT$_REST       = 0000001F        MAC$GL_LIST_IT    ********   X    04    PSC$M_NOSHR      = FFFFFFDF
INT$_SAME       = 00000009        MAC$INTOUT_1_LW   ********   X    04    PSC$M_NOVEC      = FFFFFDFF
INT$_SAVE       = 00000020        MAC$INTOUT_2_LW   ********   X    04    PSC$M_NOWRT      = FFFFFEFF
INT$_SBTTL      = 00000021        MAC$INTOUT_X      ********   X    04    PSC$M_OVR        = 00000004
INT$_SETFLAG    = 00000022        MAC$LCLSKIP       ********   X    04    PSC$M_PAGE       = 00006400
INT$_SETLONG    = 00000023        MAC$MAC_ARG_SCN   ********   X    04    PSC$M_PIC        = 00000001
INT$_SPIC       = 00000024        MAC$SKIPSP        ********   X    04    PSC$M_QUAD       = 00004C00
INT$_SPID       = 00000025        MAC$SRCUSRSYMTB   ********   X    04    PSC$M_RD         = 00000080
INT$_STIB       = 00000026        MAC$SRC_LIST      ********   X    04    PSC$M_REL        = 00000008
INT$_STIL       = 00000028        MAC$SYMSCNUP      ********   X    04    PSC$M_SHR        = 00000020
INT$_STIW       = 00000027        MAC$_IFDIRSYNX  = 007D9092              PSC$M_USR        = FFFFFFFD
INT$_STKEPT     = 00000029        MAC$_IFEXPRNABS=  007D909A              PSC$M_VEC        = 00000200
INT$_STKG       = 0000002A        MAC$_IFLEVLXCED=  007D90A2              PSC$M_WORD       = 00004400
INT$_STKL       = 0000002B        MAC$_ILLIFCOND  = 007D90DA              PSC$M_WRT        = 00000180
INT$_STKPC      = 0000002C        MAC$_NOTINANIF  = 007D9182              PSC$S_ALIGNMENT=   00000004
INT$_STKS       = 0000002D        MAC$_UNTERMCOND=  007D9232              PSC$V_ALIGNFLG   = 0000000E
INT$_STOB       = 00000034        MAC_SUBSYS      = 0000007D              PSC$V_ALIGNMENT=   0000000A
INT$_STOL       = 0000002E        MNB$B_ARGCT       00000017              PSC$V_EXE        = 00000006
INT$_STOW       = 00000035        MNB$B_NAME        00000004              PSC$V_GBL        = 00000004
INT$_STRB       = 0000002F        MNB$K_BLKSIZ      0000001C              PSC$V_LIB        = 00000001
INT$_STRL       = 00000031        MNB$L_ARGP        00000018              PSC$V_OVR        = 00000002
INT$_STRSB      = 00000032        MNB$L_CRSYMF      00000013              PSC$V_PIC        = 00000000
INT$_STRSW      = 00000033        MNB$L_LINK        00000000              PSC$V_RD         = 00000007
INT$_STRW       = 00000030        MNB$L_PAGC        0000000F              PSC$V_REL        = 00000003
INT$_STSB       = 00000036        MNB$L_PAGP        0000000B              PSC$V_SHR        = 00000005
INT$_STSW       = 00000037        MNB$L_TXTP        00000005              PSC$V_VEC        = 00000009
INT$_SUB        = 0000000A        MNB$W_FLAG        00000009              PSC$V_WRT        = 00000008
INT$_SUME       = 00000039        MXB$K_BLKSIZ      00000008              PSC$W_FLAG         00000009
```

```
PSC$W_OPTIONS     0000000D              X2             = 0000000F
RDX$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
SCAN_FALSE_CODE   0000009D R      04
SEMI            = 0000003B
STB$K_PG_MISS   = 0000000A
SYM$B_NAME        00000004
SYM$B_SEG         0000000C
SYM$B_TOKEN       0000000B
SYM$K_BLKSIZ      0000000D
SYM$K_MAXLEN    = 0000001F
SYM$K_TWOCOL    = 00000010
SYM$L_LINK        00000000
SYM$L_VAL         00000005
SYM$M_ABS       = 00000010
SYM$M_ASN       = 00000100
SYM$M_CRFO      = 00002000
SYM$M_DEBUG     = 00000020
SYM$M_DEF       = 00000001
SYM$M_DELMAC    = 00000200
SYM$M_EPT       = 00000200
SYM$M_EXTRN     = 00000008
SYM$M_GLOBL     = 00000004
SYM$M_LOCAL     = 00000040
SYM$M_ODBG      = 00000400
SYM$M_REF       = 00000080
SYM$M_RELPSECT  = 00000800
SYM$M_SUPR      = 00004000
SYM$M_WEAK      = 00000002
SYM$M_XCRF      = 00001000
SYM$V_ABS       = 00000004
SYM$V_ASN       = 00000008
SYM$V_CRFO      = 0000000D
SYM$V_DEBUG     = 00000005
SYM$V_DEF       = 00000000
SYM$V_DELMAC    = 00000009
SYM$V_EPT       = 00000009
SYM$V_EXTRN     = 00000003
SYM$V_GLOBL     = 00000002
SYM$V_LOCAL     = 00000006
SYM$V_ODBG      = 0000000A
SYM$V_REF       = 00000007
SYM$V_RELPSECT  = 0000000B
SYM$V_SUPR      = 0000000E
SYM$V_WEAK      = 00000001
SYM$V_XCRF      = 0000000C
SYM$W_FLAG        00000009
TAB             = 00000009
TRUE_FALSE        00000116 R      04
X1              = 00000400
```

```
                                             +----------------+
                                             ! 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
$ABS$                           0000001C  (    28.)  02 (   2.)  NOPIC    USR    CON    ABS    LCL NOSHR   EXE   RD     WRT NOVEC BYTE
MAC$RO_DATA                     00000203  (   515.)  03 (   3.)  NOPIC    USR    CON    REL    GBL NOSHR NOEXE   RD   NOWRT NOVEC LONG
MAC$RO_CODE_P1                  00000305  (   773.)  04 (   4.)  NOPIC    USR    CON    REL    GBL NOSHR   EXE   RD   NOWRT NOVEC LONG

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

Phase                    Page faults    CPU Time       Elapsed Time
-----                    -----------    --------       ------------
Initialization                   29    00:00:00.04    00:00:01.85
Command processing              103    00:00:00.37    00:00:06.52
Pass 1                          226    00:00:03.94    00:00:20.24
Symbol table sort                 0    00:00:00.46    00:00:01.72
Pass 2                          127    00:00:01.20    00:00:06.62
Symbol table output              34    00:00:00.19    00:00:00.24
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            523    00:00:06.22    00:00:37.21

The working set limit was 1500 pages.
38372 bytes (75 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 474 non-local and 28 local symbols.
604 source lines were read in Pass 1, producing 23 object records in Pass 2.
15 pages of virtual memory were used to define 14 macros.

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

Macro library name                          Macros defined
------------------                          --------------
_$255$DUA28:[MACRO.OBJ]MACRO.MLB;1                 12
_$255$DUA28:[SYSLIB]STARLET.MLB;2                   3
TOTALS (all libraries)                             15

546 GETS were required to define 15 macros.

There were no errors, warnings or information messages.

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

0223 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

DEFINE
MAR

LOGINCMD
LIS

ACTONE
LIS

MESSAGES
LIS

ACTCHR
LIS

VALIDATE
LIS

ACTIF
LIS

PROTCLI
LIS

CRFMACROS
MAR

MACRO

MACRO32
MAP