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

FILEID**DEFINE

```
DDDDDDDD    EEEEEEEEEE  FFFFFFFFFF    IIIIII    NN      NN  EEEEEEEEEE
DDDDDDDD    EEEEEEEEEE  FFFFFFFFFF    IIIIII    NN      NN  EEEEEEEEEE
DD      DD  EE          FF              II      NN      NN  EE
DD      DD  EE          FF              II      NN      NN  EE
DD      DD  EE          FF              II      NNNN    NN  EE
DD      DD  EE          FF              II      NNNN    NN  EE
DD      DD  EEEEEEEE    FFFFFFFF        II      NN  NN  NN  EEEEEEEE
DD      DD  EEEEEEEE    FFFFFFFF        II      NN  NN  NN  EEEEEEEE
DD      DD  EE          FF              II      NN    NNNN  EE
DD      DD  EE          FF              II      NN    NNNN  EE
DD      DD  EE          FF              II      NN      NN  EE              ....
DD      DD  EE          FF              II      NN      NN  EE              ....
DDDDDDDD    EEEEEEEEEE  FF            IIIIII    NN      NN  EEEEEEEEEE      ....
DDDDDDDD    EEEEEEEEEE  FF            IIIIII    NN      NN  EEEEEEEEEE      ....


LL              IIIIII      SSSSSSSS
LL              IIIIII      SSSSSSSS
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
```

```
:++
: E
: D
: T
:--
```

```
OPD
OPD
OPD
OPD
OPD
OPD
OPD
OPD
OPD
OPD
OPD
OPD
OPD
OPD
OPD
OPD
OPD
OPD
OPD
OPD
OPD

B
W
L
F
Q
D
G
O
H

AB=
AW=
AL=
AF=
AD=
AQ=
AG=
AO=
AH=
RB=
RW=
RL=
RF=
RD=
RQ=
RG=
RO=
RH=
MB=
MW=
```

```
        .NLIST
        .TITLE  MAC$DEFINE      MACRO LIBRARY DEFINITION FILE FOR MACRO
        .IDENT  'V04-000'
```



```
;++
; FACILITY:     VAX MACRO ASSEMBLER OBJECT LIBRARY -- FACILITY NUMBER 125.
;
; ABSTRACT:
;
; The VAX-11 MACRO assembler translates MACRO-32 source code into object
; modules for input to the VAX-11 LINKER.
;
; ENVIRONMENT:
;
; AUTHOR: Benn Schreiber, CREATION DATE: 20-AUG-78
;
; MODIFIED BY:
;
;       V03-002 ROP0024         Robert Posniak  18-JUL-1984
;               Add new flag (FLG$V_NULCHR) to indicate null
;               parameter in IRP definition.
;
;       V03-001 MTR0031         Mike Rhodes     13-Apr-1983
;               Add the .LINK directive terminal definition to the macro
;               $MAC_GRAMMARDEF.  Additionally, $MAC_TIRCMDDEF has been
;               deleted and $MAC_OBJCODDEF has been modified to include
;               the $OBJDEF macro (from SYS$LIBRARY:STARLET.MLB).
;
;       V03.00  MTR0003         Mike Rhodes     15-Mar-1982
;               Added two new flags to allow Macro to exit with the
;               proper exit status.  The flags FLG$V_EXTERR and FLG$V_EXTWRN
;               are set to indicate that a non-assembly error (eg. RMS, LBR,
;               SUM, etc.) has occurred.  Fixes SPR #11-41651(A).
;
;       V02.21  MTR0001         Mike Rhodes     02-Feb-1982
;               Add FLG$V_DLIMSTR to correct the handling of "-;"
;               strings in delimited .ASCIx strings.  QAR #890 and
;               SPR #11-42904.
```

V02.20 PCG0008 Peter George 26-Aug-1981
Add RDXV_DOUBLE, RDXV_GFLOAT, RDX$V_HFLOAT.
Correct G-Float symbol generation bug.

V02.19 PCG0002 Peter George 16-Apr-1981
Add new flag, DBGOUT, to output debugger records for abs psects. Also rename GENERIC symbol flag bit to RELPSECT.

V02.18 CNH0047 Chris Hume 22-Dec-1980
Count null argument after trailing comma for .NARG directive.
(ARGSCN 02.08, GETARG.MAR 02.06)

V02.17 CNH0042 Chris Hume 28-Oct-1980
De-optimize boundary valued backward references if indexing requested. Allow the architecturally legal immediate mode in addre , and vield contexts and also the practically useless indexed immediate mode.
(ACTREF.MAR 02.15, ACTSTA.MAR 02.15, SYMTAB.MAR 02.18)

V01.16 RN0030 R. Newland 14-Mar-1980
Increase resultant file specification buffer in macro library file block.

V01.15 RN0023 R. Newland 2-Nov-1979
Get error messages from system message file.

V01.14 RN0014 R. Newland 10-Oct-1979
Support for G_floating, H_floating and Octaword data types

V01.13 RN0013 R. Newland 27-Sep-1979
Add VEC attribute name to PSECT options

V01.12 RN0011 R. Newland 11-Sep-1979
New librarian support - redefine MLF block and remove definition of MLB block.

V01.11 RN0010 R. Newland 5-Sep-1979
Multipage MXB blocks

V01.10 RN0008 R. Newland 29-Aug-1979
31 character symbols

V01.09 RN0005 R. Newland 09-Aug-1979
Variable sized symbol name storage and symbolically defined maximum argument length.

V01.08 RN0002 R. Newland 01-Feb-1979
Changes for Source Update Merge, add global flag to mark input file as being updated, and define new intermediate code for SUM errors.
V01.16 RN0018 R. Newland 20-Oct-1979
Define bit to get and convert macro argument to upper case.

V01.15 RN0016 R. Newland 19-Oct-1979
Don't output error messages when .NTYPE operand argument is the PC. SPR 11-26392

V01.06 0003 B. Schreiber 9-JAN-1979
Catch syntax error if pound sign missing from immediate ASCII (^A) operands.

;--

KAD
KBY
KDO
KFI
KFL
KLO
KQU
KWO
KBL
KBL
KBL
KBL
KBL
KBL
KBL
KBL
KIF
KII
KIF
KIF
KIF
KIR
KIR
KRE
KEN
KEN
KMA
KMC
KME
KEN
KMD
KDE
KDS
KEN
KEN
KEN
KVE
KAL
KEV
KOD
KEX
KGL
KIN
KLI
KLI
KNL
KNA
KNC
KPA
KPS
KRE
KSA
KTI
KID
KSB
KWE
KRE
KRE
KRE
KRE
KER
KPR
KWA
KNT
KOP

```
        .SBTTL  DECLARATIONS
;
; INCLUDE FILES:
;

;
; MACROS:
;

;
; EQUATED SYMBOLS:
;

;
; OWN STORAGE:
;
```

```
        .SBTTL  SYMBOL_BLOCK DEFINITIONS

;++
;       THIS MACRO DEFINES THE SYMBOL AND PSECT BLOCK OFFSETS
;--

        .MACRO  $MAC_SYMBLKDEF

SYM$K_MAXLEN    =       31.             ; Maximum symbol name length

;++
;       ALPHA-NUMERIC SYMBOL BLOCK
;
;       THIS IS THE BASIC SYMBOL BLOCK USED TO INSERT USER-DEFINED
;       SYMBOLS INTO THE USER SYMBOL TABLE.  THE OFFSETS ARE DEFINED
;       IN RELATION TO THE BEGINNING OF THE SYMBOL BLOCK.
;--

        .PSECT  $ABS$,ABS

.=0

SYM$L_LINK:     .BLKL   1               ;LINK TO NEXT SYMBOL OR 0
SYM$B_NAME:     .BLKB   1               ;OFFSET FROM BASE TO NAME
SYM$L_VAL:      .BLKL   1               ;SYMBOL VALUE (32 BITS)
SYM$W_FLAG:     .BLKW   1               ;SYMBOL FLAGS
SYM$B_TOKEN:    .BLKB   1               ;TOKEN VALUE
SYM$B_SEG:      .BLKB   1               ;SEGMENT (PSECT) DEFINED IN
SYM$K_BLKSIZ:                           ;FIXED PART SIZE OF A SYMBOL BLOCK

;++
;       LOCAL SYMBOL (FORM nnnnn$)
;
;       SYMBOL BLOCK IS THE SAME AS A A/N SYMBOL BLOCK.  THE SYMBOL
;       NAME IS STORED AS <1 BYTE>LSB NUMBER + <2 BYTES> VALUE.
;--

;++
;       OPCODE SYMBOL
;
;       OPCODES ARE STORED IN A SEPARATE SYMBOL TABLE.  THE SYMBOL
;       BLOCK TO DEFINE AN OPCODE IS THE SAME AS THE A/N SYMBOL
;       DEFINITION BLOCK WITH THE FOLLOWING EXCEPTIONS:
;
;       1) SYM$B_SEG    DEFINES THE NUMBER OF OPERANDS WHICH THE
;                       SYMBOL MAY HAVE.  IF THE SYMBOL IS A GENERIC
;                       SYMBOL (I.E. DIV AS OPPOSED TO DIV2 OR DIV3)
;                       THIS COUNT WILL BE 0 AND A FLAG (SYM$V_GENERIC) WILL
;                       BE SET IN THE SYM$L_FLAG WORD.
;
;       2) FOLLOWING THE SYM$B_SEG BYTE THERE ARE STORED THE OPERAND
;                       BYTE DESCRIPTORS.  THERE IS ONE BYTE FOR EACH
;                       OPERAND THAT AN OPCODE MAY HAVE.
;--

;++
;       SYMBOL FLAGS
;
;       THE FOLLOWING FLAGS ARE DEFINED AND MAY BE PRESENT IN THE
;       FLAGS WORD (SYM$L_FLAG).
;--
```

```
        .MACRO  $SYM_BITDEF     SYMBOL
        SYM$M_'SYMBOL=X1
        SYM$V_'SYMBOL=X2
        X1 = X1 @ 1
        X2=X2+1
        .ENDM   $SYM_BITDEF

        X1=1
        X2=0

                                        ; BIT ON IN FLAG WORD IMPLIES:
$SYM_BITDEF     DEF             ;SYMBOL HAS BEEN DEFINED
$SYM_BITDEF     WEAK            ;SYMBOL IS DEFINED .WEAK
$SYM_BITDEF     GLOBL           ;SYMBOL IS DEFINED .GLOBAL
$SYM_BITDEF     EXTRN           ;SYMBOL IS DEFINED .EXTERNAL
$SYM_BITDEF     ABS             ;SYMBOL IS ABSOLUTE
$SYM_BITDEF     DEBUG           ;SYMBOL HAS .DEBUG ATTRIBUTE
$SYM_BITDEF     LOCAL           ;SYMBOL IS A LOCAL LABEL
$SYM_BITDEF     REF             ;SYMBOL HAS BEEN REFERENCED
$SYM_BITDEF     ASN             ;SYMBOL CREATED BY ASSIGNMENT STATEMENT
$SYM_BITDEF     EPT             ;SYMBOL CREATED BY .ENTRY
        SYM$M_DELMAC=SYM$M_EPT  ;MACRO DEFINITION HAS BEEN DELETED
        SYM$V_DELMAC=SYM$V_EPT  ;MACRO DEFINITION HAS BEEN DELETED
$SYM_BITDEF     ODBG            ;SYMBOL TO BE OUTPUT TO DEBUG RECORD
$SYM_BITDEF     RELPSECT        ;SYMBOL IS REFERENCED IN A REL PSECT
$SYM_BITDEF     XCRF            ;DO NOT CREF THIS SYMBOL
$SYM_BITDEF     CRFO            ;INSERT KEY HAS BEEN DONE FOR THIS SYMBOL
$SYM_BITDEF     SUPR            ;BIT IS SET UNTIL SYMBOL IS REFERENCED

;++
;       PSECT NAME BLOCK
;
;       THE PSECT NAME BLOCK IS IDENTICAL TO THE A/N SYMBOL BLOCK WITH
;       THE FOLLOWING EXCEPTIONS: 1) THE SYM$B_TOKEN BYTE IS UNUSED,
;       AND 2)THERE ARE TWO ADDITIONAL FIELDS--THE OPTIONS FLAGS WORD
;       (16 BITS) AND THE CURRENT LOCATION COUNTER (32 BITS).
;
;--

.=0

PSC$L_LINK:     .BLKL   1               ;LINK TO NEXT PSECT NAME BLOCK OR 0
PSC$B_NAME:     .BLKB   1               ;OFFSET FROM BASE TO PSECT NAME
PSC$L_MAXLGTH:  .BLKL   1               ;PSECT MAXIMUM LENGTH
PSC$W_FLAG:     .BLKW   1               ;PSECT FLAGS
PSC$B_UNUSED:   .BLKB   1               ;UNUSED BYTE
PSC$B_SEG:      .BLKB   1               ;PSECT SEGMENT NUMBER
PSC$W_OPTIONS:  .BLKW   1               ;PSECT OPTIONS
PSC$L_CURLOC:   .BLKL   1               ;PSECT CURRENT LOCATION
PSC$K_BLKSIZ:                           ;FIXED PART SIZE OF PSECT BLOCK

;++
;       PSECT FLAGS
;
;--

        .MACRO  $PSC_BITDEF     SYMBOL
        PSC$M_'SYMBOL=X1
        PSC$M_ALLOPTNS = PSC$M_ALLOPTNS ! X1
        PSC$V_'SYMBOL=PSC$K_NO_OPTNS
        X1=X1@1
        PSC$K_NO_OPTNS=PSC$K_NO_OPTNS+1
        .ENDM
```

```
        X1=1
        PSC$K_NO_OPTNS=0
        PSC$M_ALLOPTNS=0

$PSC_BITDEF     PIC                             ;PIC CODE
        PSC$M_NOPIC=^C<PSC$M_PIC>               ;NON-PIC CODE
$PSC_BITDEF     LIB                             ;
        PSC$M_USR=^C<PSC$M_LIB>
$PSC_BITDEF     OVR
        PSC$M_CON=^C<PSC$M_OVR>
$PSC_BITDEF     REL
        PSC$M_ABS=^C<PSC$M_REL>
$PSC_BITDEF     GBL
        PSC$M_LCL=^C<PSC$M_GBL>
$PSC_BITDEF     SHR
        PSC$M_NOSHR=^C<PSC$M_SHR>
$PSC_BITDEF     EXE
        PSC$M_NOEXE=^C<PSC$M_EXE>
$PSC_BITDEF     RD
        PSC$M_NORD=^C<PSC$M_RD>
$PSC_BITDEF     WRT
        PSC$M_NOWRT=^C<PSC$M_WRT>
$PSC_BITDEF     VEC
        PSC$M_NOVEC=^C<PSC$M_VEC>
PSC$M_EXE=PSC$M_EXE+PSC$M_RD                    ;READ ACCESS FOR EXE
PSC$M_WRT=PSC$M_WRT+PSC$M_RD                    ;READ ACCESS FOR WRT ALSO
PSC$V_ALIGNMENT =       PSC$K_NO_OPTNS          ;FIRST BIT OF ALIGNMENT FIELD
PSC$S_ALIGNMENT =       4                       ;SIZE OF ALIGNMENT FIELD
PSC$V_ALIGNFLG  = PSC$V_ALIGNMENT+PSC$S_ALIGNMENT ; BIT # OF ALIGNMENT FLAG
PSC$M_ALIGNFLG  = 1@PSC$V_ALIGNFLG              ;ALIGNMENT FLAG
PSC$M_BYTE      = <0@PSC$V_ALIGNMENT>!PSC$M_ALIGNFLG    ;BYTE ALIGNED
PSC$M_WORD      = <1@PSC$V_ALIGNMENT>!PSC$M_ALIGNFLG    ;WORD ALIGNED
PSC$M_LONG      = <2@PSC$V_ALIGNMENT>!PSC$M_ALIGNFLG    ;LONG ALIGNED
PSC$M_QUAD      = <3@PSC$V_ALIGNMENT>!PSC$M_ALIGNFLG    ;QUAD ALIGNED
PSC$M_PAGE      = <9@PSC$V_ALIGNMENT>!PSC$M_ALIGNFLG    ;PAGE ALIGNED

PSC$M_DEFAULT   =       PSC$M_REL!PSC$M_WRT!PSC$M_RD!PSC$M_EXE ;DEFAUTLT PSECT OPTIONS

        .PSECT
        .MDELETE        $MAC_SYMBLKDEF, $SYM_BITDEF, $PSC_BITDEF
        .ENDM   $MAC_SYMBLKDEF
```

```
        .MACRO  $MAC_INPBLKDEF

;++
;
;       INPUT LINE CONTEXT BLOCK DEFINITIONS
;
;       THIS CONTEXT BLOCK IS USED BY THE CHARACTER INPUT ROUTINE TO
;       GET THE NEXT LINE WHEN NEEDED.
;
;--

        $MAC_MNBDEF


        .PSECT  $ABS$,ABS

.=0

INP$L_LINK:     .BLKL   1               ;LINK TO NEXT BLOCK
INP$L_NXTL:     .BLKL   1               ;LINK TO NEXT LINE
INP$L_GETL:     .BLKL   1               ;ADDRESS OF ROUTINE TO GET NEXT LINE
INP$L_IFLVL:    .BLKL   1               ;"IF" LEVEL COUNTER
INP$L_IFVAL:    .BLKL   1               ;"IF" VALUE
INP$L_RPTCNT:   .BLKL   1               ;COUNTER FOR REPEAT/IRP
INP$L_PAGP:     .BLKL   1               ;POINTER TO PAGES USED BY THIS INPUT
                                        ;BLOCK
INP$B_ARGCT:    .BLKB   1               ;ARGUMENT COUNT
INP$L_ARGS:     .BLKL   1               ;POINTERS TO ARGUMENTS
INP$K_BLKSIZ:                           ;SIZE OF AN INPUT BLOCK

.=0

                .BLKB   <MNB$K_BLKSIZ+SYM$K_MAXLEN+1>  ;SIZE WHEN MNB IS MADE INTO INP
INP$K_IRPSIZ:

        .MDELETE        $MAC_INPBLKDEF
        .ENDM   $MAC_INPBLKDEF

;++
;       MACRO NAME BLOCK
;
;       THIS MACRO DEFINES THE MACRO NAME BLOCK (MNB) AND THE MACRO
;       ARGUMENT BLOCK (MAB).  THESE ARE USED FOR DEFINING MACROS.
;
;--

        .MACRO  $MAC_MNBDEF

        .PSECT  $ABS$,ABS

.=0

MNB$L_LINK:     .BLKL   1               ;LINK TO NEXT MNB
MNB$B_NAME:     .BLKB   1               ;OFFSET FROM BASE TO MACRO NAME
MNB$L_TXTP:     .BLKL   1               ;POINTER TO TEXT
MNB$W_FLAG:     .BLKW   1               ;FLAGS
MNB$L_PAGP:     .BLKL   1               ;POINTER TO PAGES USED TO DEFINE MACRO
MNB$L_PAGC:     .BLKL   1               ;COUNT OF PAGES USED
MNB$L_CRSYMF:   .BLKL   1               ;CREATED SYMBOL FLAGS
MNB$B_ARGCT:    .BLKB   1               ;ARGUMENT COUNT (255. MAX)
MNB$L_ARGP:     .BLKL   1               ;POINTER TO ARG LIST
MNB$K_BLKSIZ:                           ;FIXED PART SIZE OF MACRO NAME BLOCK
```

```
;
; MAB -- MACRO ARGUMENT BLOCK
;

.=0

MAB$L_LINK:     .BLKL   1               ;LINK TO NEXT MAB OR 0
MAB$B_NAME:     .BLKB   1               ;OFFSET FROM BASE TO ARGUMENT NAME
MAB$B_ARGNO:    .BLKB   1               ;ARG NUMBER (1-255.)
MAB$W_DVLEN:    .BLKW   1               ;LENGTH OF DEF. VALUE STRING
MAB$L_DVPTR:    .BLKL   1               ;PTR TO DEF. VALUE STRING
MAB$K_BLKSIZ:                           ;FIXED PART SIZE OF MAB

;
; MXB -- MACRO EXTENSION BLOCK
;

.=0

MXB$L_LINK:     .BLKL   1               ; Link to next MXB
MXB$L_PAGES:    .BLKL   1               ; Number of pages in block
MXB$K_BLKSIZ:                           ; Size of MXB

        .MACRO  $MAC_MNBDEF
        .ENDM   $MAC_MNBDEF

        .ENDM   $MAC_MNBDEF
```

```
;++
;       $MAC_GENVALDEF
;
;       DEFINES THE COMMONLY USED VALUES.  MUST BE INVOKED IN ANY
;       MODULE THAT REFERENCES ANY OF THESE VALUES.
;
;--

        .MACRO  $MAC_GENVALDEF

MAC_SUBSYS =    125.                    ;MACRO-32 FACILITY NUMBER
HASHSZ  =       127.                    ;HASH TABLE SIZE
CR      =       ^O15                    ;CARRIAGE RETURN
TAB     =       ^O11                    ;HORIZONTAL TAB
FF      =       ^O14                    ;FORM FEED
BLNK    =       ^O40                    ;BLANK
HYPHEN  =       ^A/-/                   ;HYPHEN
SEMI    =       ^A/;/                   ;SEMI COLON
CHR$V_CVTLWC    =       ^A/a/           ;CONVERT ANYTHING .GT. LOWERCASE A
CHR$V_NOCVT     =       ^X7F            ;DO NOT CONVERT TO UPPERCASE
INT$K_BUFSIZ    =       <10*512>-<3*4>  ; SIZE OF INT. BUFFER
INT$K_BUFWRN    =       INT$K_BUFSIZ-100 ;WARNING LIMIT
INP$K_BUFSIZ    =       1000.           ; Input buffer size
OBJ$K_BUFSIZ    =       512             ;OBJECT FILE BUFFER SIZE
LST$K_BUFSIZ    =       134.            ;LISTING FILE BUFFER SIZE
LST$K_TITLE_SIZ =       40.             ; Size of title sub-string
REG$_PC         =       15              ;PC REGISTER
RDX$V_BINARY    =       0               ;INDEX FOR BINARY RADIX
RDX$V_OCTAL     =       1               ;INDEX FOR OCTAL RADIX
RDX$V_DECIMAL   =       2               ;INDEX FOR DECIMAL RADIX
RDX$V_HEX       =       3               ;INDEX FOR HEX RADIX
RDX$V_FLOAT     =       4               ;INDEX FOR F FLOATING NUMBERS
RDX$V_DOUBLE    =       5               ;INDEX FOR D FLOATING NUMBERS
RDX$V_GFLOAT    =       6               ;INDEX FOR G FLOATING NUMBERS
RDX$V_HFLOAT    =       7               ;INDEX FOR H FLOATING NUMBERS
LST$K_L_P_PAGE  =       60              ;LINES PER PAGE IN LISTING
STB$K_PG_MISS   =       10              ;# PAGES TO ALLOCATE FOR EMPTY SYMBOL BUCKET
SYM$K_MAXLEN    =       31              ; Maximum characters/symbol
SYM$K_TWOCOL    =       16              ; Character size for 2 column symbol table
CHR$M_SPA_MSK   =       ^X01            ;BIT MASK FOR SPACE-LIKE CHARACTER
CHR$V_SPA_MSK   =       0               ;BIT NUMBER FOR SPACE-LIKE CHAR
CHR$M_SYM_DLM   =       ^X02            ;BIT MASK FOR SYMBOL DELIMITER CHARACTER
CHR$V_SYM_DLM   =       1               ;BIT NUMBER FOR SYMBOL DELIM.
CHR$M_SYM_CHR   =       ^X04            ;BIT MASK FOR CHAR THAT CAN START SYMBOL
CHR$V_SYM_CHR   =       2               ;BIT NUMBER FOR SYMBOL CHAR
CHR$M_SYM_CH1   =       ^X08            ;BIT MASK FOR CHAR THAT CAN BE IN SYMBOL
CHR$V_SYM_CH1   =       3               ;BIT NUMBER FOR SYMBOL CHAR
CHR$M_NUM_BER   =       ^X10            ;BIT MASK FOR NUMBER
CHR$V_NUM_BER   =       4               ;BIT NUMBER FOR NUMBER
CHR$M_COMMA_CR  =       ^X20            ;BIT MASK FOR COMMA AND CR
CHR$V_COMMA_CR  =       5               ;BIT NUMBER FOR COMMA AND CR
CHR$M_ILL_CHR   =       ^X40            ;BIT MASK FOR ILLEGAL CHARACTER
CHR$V_ILL_CHR   =       6               ;BIT NUMBER FOR ILLEGAL CHARACTER
ARG$K_SIZE      =       1000            ; Maximum characters in MACRO argument
AUD$K_SIZE      =       16              ; Audit trail string size

        .MDELETE        $MAC_GENVALDEF
        .ENDM   $MAC_GENVALDEF
```

```
        .SBTTL  DEFINE MACRO TEXT SPECIAL MARKERS

;++
;       $MAC_MTXDEF
;
;       DEFINES THE MACRO TEXT SPECIAL MARKERS USED DURING MACRO
;       AND REPEAT/IRP PROCESSING.
;--

        .MACRO  $MAC_MTXDEF

MTX$_ARGMRK     =       ^XFF            ;ARG NO. FOLLOWS (BYTE)
MTX$_TXTLNK     =       ^XFE            ;LONGWORD OF LINK FOLLOWS
MTX$_LXLEN      =       ^XFD            ;LEXICAL OPERATOR--LENGTH
MTX$_LXEXT      =       ^XFC            ;LEXICAL OPERATOR--EXTRACT
MTX$_LXLOC      =       ^XFB            ;LEXICAL OPERATOR--LOCATE
;               ^XFA TO ^XF0 RESERVED
MTX$_LITSTR     =       ^XEF            ;LITERAL STRING FOLLOWS
MTX$_LITVAL     =       ^XEE            ;LITERAL VALUE FOLLOWS
MTX$_SYMADR     =       ^XED            ;SYMBOL ADDRESS FOLLOWS
MTX$_NOMORE     =       ^XEC            ;NO MORE ARGUMENTS

        .ENDM   $MAC_MTXDEF

;++
; MTX$_ARGMRK   FOLLOWED BY A BYTE CONTAINING THE ARGUMENT NUMBER
; MTX$_TXTLNK   FOLLOWED BY A LONGWORD OF ADDRESS (TEXT LINK)
; MTX$_LITSTR   FOLLOWED BY WORD OF COUNT, THEN ASCII STRING
; MTX$_LITVAL   FOLLOWED BY LONGWORD OF VALUE
; MTX$_SYMADR   FOLLOWED BY LONGWORD OF SYMBOL ADDRESS
;--
```

```
        .SBTTL  DEFINE MACRO LIBRARY FILE OFFSETS

        .MACRO  $MAC_MLFDEF

        $NAMDEF

        .PSECT  $ABS$,ABS

.=0
        MLF$K_RSFNLN    = NAM$C_MAXRSS  ; Resultant file name length

MLF$L_QLINK:    .BLKL   2               ;QUEUE LINK WORDS
MLF$L_MCDEF:    .BLKL   1               ;# MACROS DEFINED IN THIS MLB
                                        ;(I.E. IMCALL'S SATISFIED HERE)
MLF$Q_FNAMDS:   .BLKQ   1               ; File name descriptor
MLF$L_CTINDEX:  .BLKL   1               ; Control table index
MLF$X_NAMBLK:   .BLKB   NAM$C_BLN       ; NAM block
MLF$T_FNAM:     .BLKB   MLF$K_RSFNLN    ; File name string
MLF$K_BLKSIZ:

        .MDELETE        $MAC_MLFDEF
        .ENDM   $MAC_MLFDEF
```

```
        .SBTTL  DEFINE OPERAND DESCRIPTOR FLAG BITS

;++
; EACH OPCODE SYMBOL BLOCK IS FOLLOWED BY A SERIES OF WORDS WHICH
; DESCRIBE THE OPERANDS FOR THE OPCODE.  THE FOLLOWING MACRO DEFINES
; THOSE WORD DESCRIPTORS.
;
;--

        .MACRO  $MAC_OPRDEF

OPD$S_SIZE      =       5               ;SIZE OF BYTES FIELD
OPD$V_SIZE      =       0               ;POSITION OF BYTES FIELD
OPD$S_MODE      =       5               ;SIZE OF ACCESS MODE FIELD
OPD$V_MODE      =       5               ;POSITION OF ACCESS MODE FIELD
OPD$M_MODE = 1@OPD$S_MODE-1@OPD$V_MODE  ; Mask to isolate mode value
OPD$M_ADDR      =       0@<OPD$V_MODE>  ;ADDRESS ACCESS MODE
OPD$M_READ      =       1@<OPD$V_MODE>  ;READ ACCESS MODE
OPD$M_MODIFY    =       2@<OPD$V_MODE>  ;MODIFY ACCESS MODE
OPD$M_WRITE     =       3@<OPD$V_MODE>  ;WRITE ACCESS MODE
OPD$M_VIELD     =       4@<OPD$V_MODE>  ;FIELD ACCESS
OPD$M_BB        =       5@<OPD$V_MODE>+1;BRANCH BYTE
OPD$M_BW        =       6@<OPD$V_MODE>+2;BRANCH WORD
OPD$M_FLOAT     =       ^X8000          ;FLOATING POINT (MUST BE SIGN BIT)
OPD$V_FLOAT     =       15              ;BIT NUMBER FOR FLT. PT.
OPD$M_D_FLOAT   =       OPD$M_FLOAT+^X4000      ;D-FLOATING
OPD$V_D_FLOAT   =       14              ;BIT NUMBER FOR D-FLOATING
OPD$M_G_FLOAT   =       OPD$M_FLOAT+^X2000      ;G-FLOATING
OPD$V_G_FLOAT   =       13              ;BIT NUMBER FOR G-FLOATING
OPD$M_H_FLOAT   =       OPD$M_FLOAT+^X1000      ;H-FLOATING
OPD$V_H_FLOAT   =       12              ;BIT NUMBER FOR H-FLOATING
OPD$M_NOT_32F   =       ^X4000+^X2000+^X1000    ;NOT SINGLE PREC. IF THIS SET

B       =       1                       ;BYTE
W       =       2                       ;WORD
L       =       4                       ;LONGWORD
F       =       4+OPD$M_FLOAT           ;FLOATING
Q       =       8                       ;QUADWORD
D       =       8+OPD$M_D_FLOAT         ;DOUBLE FLOATING
G       =       8+OPD$M_G_FLOAT         ;G-FLOATING
O       =       16                      ;OCTA-WORD
H       =       16+OPD$M_H_FLOAT        ;H-FLOATING

AB=OPD$M_ADDR+B
AW=OPD$M_ADDR+W
AL=OPD$M_ADDR+L
AF=OPD$M_ADDR+F
AD=OPD$M_ADDR+D
AQ=OPD$M_ADDR+Q
AG=OPD$M_ADDR+G
AO=OPD$M_ADDR+O
AH=OPD$M_ADDR+H
RB=OPD$M_READ+B
RW=OPD$M_READ+W
RL=OPD$M_READ+L
RF=OPD$M_READ+F
RD=OPD$M_READ+D
RQ=OPD$M_READ+Q
RG=OPD$M_READ+G
RO=OPD$M_READ+O
RH=OPD$M_READ+H
MB=OPD$M_MODIFY+B
MW=OPD$M_MODIFY+W
```

```
ML=OPD$M_MODIFY+L
MF=OPD$M_MODIFY+F
MD=OPD$M_MODIFY+D
MQ=OPD$M_MODIFY+Q
MG=OPD$M_MODIFY+G
MO=OPD$M_MODIFY+O
MH=OPD$M_MODIFY+H
WB=OPD$M_WRITE+B
WW=OPD$M_WRITE+W
WL=OPD$M_WRITE+L
WF=OPD$M_WRITE+F
WD=OPD$M_WRITE+D
WQ=OPD$M_WRITE+Q
WG=OPD$M_WRITE+G
WO=OPD$M_WRITE+O
WH=OPD$M_WRITE+H
VB=OPD$M_VIELD+B
VW=OPD$M_VIELD+W
VL=OPD$M_VIELD+L
VF=OPD$M_VIELD+F
VD=OPD$M_VIELD+D
VQ=OPD$M_VIELD+Q
VG=OPD$M_VIELD+G
VO=OPD$M_VIELD+O
VH=OPD$M_VIELD+H

        .ENDM   $MAC_OPRDEF
```

```
.SB

.MA
.IF
MOV
.EN
BSB
.EN

.SB

.MA
MOV
BSB
```

```
        .SBTTL  TERMINAL GRAMMAR SYMBOL DEFINITIONS

;++
;       THIS MACRO DEFINES THE TERMINAL GRAMMAR SYMBOLS IN THE
;       VAX-11 MACRO GRAMMAR
;--

        .MACRO  $MAC_GRAMMARDEF

ERR01           =       1.
ERR02           =       2.
ERR03           =       3.
ERR04           =       4.
ERR05           =       5.
ERR06           =       6.
ERR07           =       7.
ERR08           =       8.
ERR09           =       9.
GOALSY          =       10.
DEOL            =       11.
ID              =       12.
MACTXT          =       13.
DOPCODE         =       14.
DCOMMA          =       15.
DCOLON          =       16.
DEQ             =       17.
DPC             =       18.
DSQOPN          =       19.
DSQCLS          =       20.
DANGOPN         =       21.
DANGCLS         =       22.
DOPN            =       23.
DCLS            =       24.
DPLUS           =       25.
DMINUS          =       26.
DTIMES          =       27.
DDIV            =       28.
DAND            =       29.
DOR             =       30.
DXOR            =       31.
DAT             =       32.
DPOUND          =       33.
DINTEGER        =       34.
DUPA            =       35.
DUPB            =       36.
DUPC            =       37.
DUPD            =       38.
DUPO            =       39.
DUPF            =       40.
DUPM            =       41.
DUPX            =       42.
DBUP            =       43.
DGUP            =       44.
DIUP            =       45.
DLUP            =       46.
DSUP            =       47.
DWUP            =       48.
RRREG           =       49.
DMASK           =       50.
KASCIC          =       51.
KASCII          =       52.
KASCIZ          =       53.
KPACKED         =       54.
```

```
KADDRESS        =       55.
KBYTE           =       56.
KDOUBLE         =       57.
KFIELD          =       58.
KFLOAT          =       59.
KLONG           =       60.
KQUAD           =       61.
KWORD           =       62.
KBLKA           =       63.
KBLKB           =       64.
KBLKD           =       65.
KBLKF           =       66.
KBLKL           =       67.
KBLKQ           =       68.
KBLKW           =       69.
KIF             =       70.
KIIF            =       71.
KIFF            =       72.
KIFT            =       73.
KIFTF           =       74.
KIRP            =       75.
KIRPC           =       76.
KREPT           =       77.
KENDC           =       78.
KENDR           =       79.
KMACRO          =       80.
KMCALL          =       81.
KMEXIT          =       82.
KENDM           =       83.
KMDELETE        =       84.
KDEBUG          =       85.
KDSABL          =       86.
KENABL          =       87.
KENTRY          =       88.
KVECTOR         =       89.
KALIGN          =       90.
KEVEN           =       91.
KODD            =       92.
KEXTRN          =       93.
KGLOBL          =       94.
KINCLUDE        =       95.
KLIBRARY        =       96.
KLIST           =       97.
KNLIST          =       98.
KNARG           =       99.
KNCHR           =       100.
KPAGE           =       101.
KPSECT          =       102.
KRESTORE        =       103.
KSAVE           =       104.
KTITLE          =       105.
KIDENT          =       106.
KSBTTL          =       107.
KWEAK           =       108.
KREF1           =       109.
KREF2           =       110.
KREF4           =       111.
KREF8           =       112.
KERROR          =       113.
KPRINT          =       114.
KWARN           =       115.
KNTYPE          =       116.
KOPDEF          =       117.
```

```
KEND                =       118.
KXFER               =       119.
KASCID              =       120.
KCROSS              =       121.
KNCROS              =       122.
KDFLT               =       123.
KSGNB               =       124.
KSGNW               =       125.
KBLKG               =       126.
KBLKH               =       127.
KBLKO               =       128.
KGFLOAT             =       129.
KHFLOAT             =       130.
KOCTA               =       131.
KREF16              =       132.
KLINK               =       133.

        .MDELETE        $MAC_GRAMMARDEF
        .ENDM   $MAC_GRAMMARDEF
```

**F

```
        .SBTTL  FLAG DEFINITIONS

;++
;       THIS MACRO DEFINES THE BIT FLAGS THAT LIVE IN THE FLAGS WORD
;       'MAC$FLAGS'.  A 0 IN A BIT POSITION INDICATES FALSE AND A 1
;       INDICATES TRUE.
;--

        .MACRO  $MAC_CTLFLGDEF
        .MACRO  $FLG_DEFINE     F
        FLG$V_'F = X1
        FLG$M_'F = X2
        X1 = X1 + 1
        X2 = X2 @ 1
        .ENDM   $FLG_DEFINE

        X1=0
        X2=1

                                        ;TRUE INDICATES:
        $FLG_DEFINE     ALLCHR          ;MAC$GETCHR SHOULD PASS SEMICOLONS
                                        ;(ALLCHR MUST BE LOW BIT!!!)
        $FLG_DEFINE     BOL             ;OPCODE NOT YET FOUND
        $FLG_DEFINE     COMPEXPR        ;EXPRESSION IS COMPILE TIME
        $FLG_DEFINE     CONT            ;ALLOW LINE CONTINUATIONS
        $FLG_DEFINE     DATRPT          ;REPEATING DATA DEFINITION
        $FLG_DEFINE     ENDMCH          ;CHECK MACRO DIRECTIVES
        $FLG_DEFINE     EVALEXPR        ;EXPR TO BE EVAL ON PASS 2
        $FLG_DEFINE     EXPOPT          ;EXPRESSION CAN BE OPTIMIZED
        $FLG_DEFINE     INSERT          ;INSERT SYMBOL IF NOT FOUND
        $FLG_DEFINE     LSTXST          ;LISTING FILE EXISTS
        $FLG_DEFINE     NEWPND          ;NEW PAGE PENDING
        $FLG_DEFINE     MACL            ;MACRO TEXT LINE PRESENT
        $FLG_DEFINE     MEBLST          ;SPECIAL LIST MEB FLAG
        $FLG_DEFINE     OPRND           ;SCANNING OPERAND FIELD
        $FLG_DEFINE     P2              ;IN PASS 2
        $FLG_DEFINE     SKAN            ;ALLOW SCANNING
        $FLG_DEFINE     MACTXT          ;READING FROM MACRO TEXT
                                        ;(FALSE INDICATES READING SOURCE FILE)
        $FLG_DEFINE     ORDLST          ;SYMBOL TABLE LIST IS ORDERED
                                        ;(OPCODES ARE NOT ORDERED IN BUCKETS)
        $FLG_DEFINE     STOIMF          ;STORING IMMEDIATE DATA (MAC$STOIM)
        $FLG_DEFINE     TOCFLG          ;PAGE HEADER FOR SUBTITLES NOT NEEDED
        $FLG_DEFINE     CHKLPND         ;THERE IS A CHKL PENDING FROM A CONTINUED LINE
        $FLG_DEFINE     OBJXST          ;THERE IS AN OBJECT FILE
        $FLG_DEFINE     IIF             ;THIS IS .IIF AS OPPOSED TO .IF
        $FLG_DEFINE     IFSTAT          ;THIS IS AN IF STATEMENT
        $FLG_DEFINE     NOREF           ;DO NOT SET REF WHEN SYMBOL IS FOUND
                                        ;(USED ONLY BY .NTYPE NOW)
        $FLG_DEFINE     SEQFIL          ;INPUT FILE HAS SEQUENCE NUMBERS
        $FLG_DEFINE     SPLALL          ;SPECIAL PASS ALL CHARACTERS FOR
                                        ;READING CERTAIN DIRECTIVES WHILE
                                        ;DEFINING MACROS
        $FLG_DEFINE     MACLTB          ;THERE IS MACRO TEXT IN BUFFER (P2)
        $FLG_DEFINE     RPTIRP          ;IN AN IRP OR REPEAT (PASS 1)
        $FLG_DEFINE     IRPC            ;IRPC AS OPPOSED TO IRP
        $FLG_DEFINE     CRF             ;/CROSS REQUESTED
        $FLG_DEFINE     XCRF            ;.NOCROSS IN EFFECT

        X2=1                            ;**INTO SECOND FLAGS WORD!!
        $FLG_DEFINE     CRSEEN          ;FLAG FOR TOKEN THAT CR WAS SEEN ONCE.
        $FLG_DEFINE     LEXOP           ;ARGSCN CALLED FOR LEXICAL OPERATOR
        $FLG_DEFINE     SPECOP          ;SPECIAL OPERATOR OUTPUT FOR LINE (MAC$BDYSCN)
```

```
        $FLG_DEFINE     MOREINP         ;MORE INPUT FILES COMING
        $FLG_DEFINE     UPAFLG          ;SET WHEN ^A/text/ SEEN
        $FLG_DEFINE     NTYPEPC         ; Suppress PC check for .NTYPE
        $FLG_DEFINE     UPMARG          ; Set to get Macro argument upper cased
        $FLG_DEFINE     UPDFIL          ; Input file is being updated
        $FLG_DEFINE     OPNDCHK         ; Operand check performed
        $FLG_DEFINE     FIRSTLN         ; First listing line
        $FLG_DEFINE     SYM2COL         ; Two column symbol table listing
        $FLG_DEFINE     MAC2COL         ; Two column macro listing
        $FLG_DEFINE     OPTVFLIDX       ; Optimized ref. will oVFL if indexed
        $FLG_DEFINE     MOREARG         ; Previous argument had trailing comma
        $FLG_DEFINE     DBGOUT          ; Output debugger records for abs psectss
        $FLG_DEFINE     DLIMSTR         ; Pass ALL characters in delimited .ASCIx strings
        $FLG_DEFINE     EXTERR          ; An external (non-assembly) error has occurred.
        $FLG_DEFINE     EXTWRN          ; An external (non-assembly) warning has occurred.
        $FLG_DEFINE     NULCHR          ; Null character as a parameter

;
; FLAGS IN MAC$GL_OPSIZE
;
        OPF$M_LASTOPR   =       ^X2000  ;LAST OPERAND IN OPCODE
        OPF$V_LASTOPR   =       13      ;
        OPF$M_OPTEXP    =       ^X1000  ;EXPRESSION HAS BEEN OPTIMIZED
        OPF$V_OPTEXP    =       12      ;

        .MDELETE        $MAC_CTLFLGDEF, $FLG_DEFINE
        .ENDM   $MAC_CTLFLGDEF
```

```
        .SBTTL  CRF_FLG FLAG BITS

        .MACRO  $MAC_CRFLAGDEF

CRF$M_DIR               =       1               ;CREF DIRECTIVES
CRF$V_DIR               =       0               ;
CRF$M_MACROS            =       2               ;CREF MACROS
CRF$V_MACROS            =       1               ;
CRF$M_OPCODES           =       4               ;CREF OPCODES
CRF$V_OPCODES           =       2               ;
CRF$M_REGISTERS         =       8               ;CREF REGISTERS
CRF$V_REGISTERS         =       3               ;
CRF$M_SYMBOLS           =       16              ;CREF SYMBOLS
CRF$V_SYMBOLS           =       4               ;

CRF$M_DEFAULT           =       CRF$M_MACROS!CRF$M_SYMBOLS

        .ENDM   $MAC_CRFLAGDEF
```

```
        .SBTTL  ADDRESSING MODE DEFINITIONS

;++
;       THIS MACRO DEFINES THE ADDRESSING MODES.
;--

.MACRO  $MAC_ADRMODDEF
X=0
.MACRO  COD     SYM
ADM$_'SYM=X
X=X+1
.ENDM

COD     LITERAL         ;LITERAL MODE
COD     IMMEDIATE       ;IMMEDIATE MODE
COD     ABSOLUTE        ;ABSOLUTE
COD     PIC             ;PIC
COD     INDEX           ;E[R]
COD     REGISTER        ;R
COD     RRIND           ;(R)
COD     REGAUTODEC      ;-(R)
COD     REGAUTOINC      ;(R)+
COD     DFRAUTOINC      ;@(R)+
COD     BYTE_DISP       ;B^D(R)
COD     DFBYTEDISP      ;@B^D(R)
COD     WORD_DISP       ;W^D(R)
COD     DFWORDDISP      ;@W^D(R)
COD     LONG_DISP       ;L^D(R)
COD     DFLONGDISP      ;@L^D(R)
        ADM$_MAXMOD     =       X-1     ;MAX. ALLOWABLE MODE

.MDELETE        $MAC_ADDRMODDEF, COD
.ENDM   $MAC_ADRMODDEF
```

```
        .SBTTL  INTERMEDIATE FILE OPCODE DEFINITIONS

;++
; PASS 1 OF THE ASSEMBLER GENERATES AN INTERMEDIATE CODE FILE
; WHICH IS PROCESSED BY PASS 2 TO PRODUCE THE OBJECT FILE.
; THE INTERMEDIATE CODE HAS THE FOLLOWING FORMAT:
;
;       ******************
;       *    LENGTH      *       LENGTH IN BYTES OF THIS FRAME
;       ******************       1 BYTE
;       *    ACTION      *       ACTION TO PERFORM (SEE CODES BELOW)
;       ******************       1 BYTE
;       *                *
;       *     DATA       *       DATA ASSOCIATED WITH FRAME
;       *                *
;       ******************       VARIABLE LENGTH
;
;
; THE MACRO BELOW DEFINES THE ACTION CODES, AND GENERATES A BRANCH TABLE.
;       THIS MACRO SHOULD BE INVOKED WITH NO ARGUMENT TO DEFINE THE
;       SYMBOLS 'INT$_SYMBOL', AND INVOKED WITH THE ARGUMENT
;       'DISPATCHTABLE' TO GENERATE THE PASS2 DISPATCH TABLE.
;--
        .MACRO  $MAC_INTCODDEF  TYPE

        $COUNT  =       0

        .IF DIF <TYPE>,<DISPATCHTABLE>


        .MACRO  INT     SYM
        INT$_'SYM = $COUNT
        $COUNT = $COUNT + 1
        .ENDM

        .IFF

        .MACRO  INT     SYM
        .LONG   P2$'SYM
        $COUNT=$COUNT+1
        .ENDM

        .ENDC

        INT     ILG     ;ILLEGAL INTERMEDIATE CODE (0)
;***BELOW SYMBOLS ARE ARITHMEITIC OPERATORS***POSITION MUST NOT CHANGE
        INT     ADD     ;ADD TOP TWO ITEMS ON STACK
        INT     AND     ;LOGICAL AND
        INT     ASH     ;ASH
        INT     DIV     ;DIVIDE
        INT     MUL     ;MULTIPLY
        INT     NEG     ;NEGATE
        INT     NOT     ;LOGICAL NOT
        INT     OR      ;LOGICAL OR
        INT     SAME    ;UNARY PLUS
        INT     SUB     ;SUBTRACT
        INT     XOR     ;EXCLUSIZE OR
;***ABOVE SYMBOLS ARE ARITHMETIC OPERATORS***POSITION MUST NOT CHANGE
        INT     ASN     ;ASSIGNMENT
        INT     AUGPC   ;AUGMENT PC
```

```
INT     BDST    ;GENERATE BRANCH DESTINATION
INT     CHKL    ;ALLIGN OUTPUT WITH SOURCE LINES
INT     END     ;END OF INTERMEDIATE CODE
INT     EPT     ;ENTRY POINT DEFINITION
INT     ERR     ;PASS 1 ERROR MESSAGE
INT     ETX     ;ERROR TEXT FOR .ERROR/.WARN/.PRINT
INT     FNEWL   ;FORCE NEW LISTING LINE
INT     LGLAB   ;STANDARD (NOT LOCAL) LABEL
INT     MACL    ;MACRO TEXT LINE
INT     NEWL    ;NEW LINE OF SOURCE TEXT
INT     NEWP    ;DEFINE NEW PSECT
INT     OP      ;GENERATE OPCODE
INT     PRIL    ;PRINT LONG WORD IN LISTING
INT     PRT     ;PASS 1 PRINT DIRECTIVE
INT     PSECT   ;CHANGE PSECT
INT     REDEF   ;.TRANSFER DIRECTIVE
INT     REF     ;GENERATE REFERENCE
INT     REST    ;RESTORE PSECT
INT     SAVE    ;SAVE PSECT
INT     SBTTL   ;SUBTITLE
INT     SETFLAG ;SET FLAG IN MAC$GL_FLAGS
INT     SETLONG ;STORE LONGWORD VALUE
INT     SPIC    ;STORE POSITION INDEPENDENT (NOT USED AT PRESENT)
INT     SPID    ;STORE ADDRESS
INT     STIB    ;STORE IMMEDIATE BYTE
INT     STIW    ;STORE IMMEDIATE WORD
INT     STIL    ;STORE IMMEDIATE LONGWORD
INT     STKEPT  ;STACK ENTRY POINT
INT     STKG    ;STACK GLOBAL SYMBOL
INT     STKL    ;STACK LITERAL VALUE
INT     STKPC   ;STACK CURRENT PC
INT     STKS    ;STACK SYMBOL
INT     STOL    ;STORE LONG WORD
INT     STRB    ;STORE REPEATED BYTE
INT     STRW    ;STORE REPEATED WORD
INT     STRL    ;STORE REPEATED LONG
INT     STRSB   ;STORE REPEATED SIGNED BYTE
INT     STRSW   ;STORE REPEATED SIGNED WORD
INT     STOB    ;STORE BYTE
INT     STOW    ;STORE WORD
INT     STSB    ;STORE SIGNED BYTE
INT     STSW    ;STORE SIGNED WORD
INT     WRN     ;PASS 1 WARNING MESSAGE
INT     SUME    ;Source-update-merge error
INT     INFO    ;Information message

.MDELETE        $MAC_INTCODDEF, INT
.ENDM   $MAC_INTCODDEF
```

```
        .SBTTL  OBJECT CODE DEFINITIONS

;++
;
;       THESE SYMBOLS AND FLAG DEFINITIONS ARE SUPPLIMENTAL TO THOSE DEFINED
;       BY $OBJDEF AND ARE USED BY THE PASS 2 ACTION ROUTINES.
;
;--

        .MACRO  $MAC_OBJCODDEF

        $OBJDEF                                 ; DEFINE THE OBJECT CODE.

;
; SYMBOL DEFINITION MASK BITS
;
        SYM$F_WEAK      =       ^0001           ;       WEAK RESOLUTION
        SYM$F_DEF       =       ^0002           ;       DEFINITION
        SYM$F_UNI       =       ^0004           ;       UNIVERSAL
        SYM$F_REL       =       ^0010           ;       RELATIVE
        SYM$F_VALIDATE  =       ^0020           ;       VALIDATE

;
; END OF MODULE RECORD STATUS
;
        OBJ$C_EOM_OK    =       EOM$C_SUCCESS   ;       NO ERRORS
        OBJ$C_EOM_WRN   =       EOM$C_WARNING   ;       WARNINGS
        OBJ$C_EOM_ERR   =       EOM$C_ERROR     ;       SERIOUS ERRORS
        OBJ$C_EOM_ABORT =       EOM$C_ABORT     ;       ABORT LINK

;
; TEXT, INFORMATION AND RELOCATION
;
        TIR$C_STO_LW    =       TIR$C_STO_L     ;       STORE LONG WORD DATUM

        .MDELETE        $MAC_OBJCODDEF
        .ENDM   $MAC_OBJCODDEF
```

```
;++
; FUNCTIONAL DESCRIPTION:
;
;       THIS MACRO, $INTOUT_X, IS USED WHEN THERE IS NEED TO ONLY
;       PUT AN ACTION INTO THE INTERMEDIATE FILE. FOR EXAMPLE,
;       INT$_CHKL, THE ALIGN SOURCE DIRECTIVE, TAKES NO ARGUMENTS.
;
;--

        .MACRO  $INTOUT_X       ACTN
        MOVZBL  #ACTN,R0
        BSBW    MAC$INTOUT_X
        .ENDM
```

```
.MACRO  $INTOUT_LW      ACTN,DATA
INTOUTC2 DATA
MOVZBL  #ACTN,R0
ERR=0
.IF IDN <AC'N>,<INT$_ERR>
ERR=1
.ENDC
.IF IDN <ACTN>,<INT$_WRN>
ERR=1
.ENDC
INT_LW_CALL \CNT
.ENDM

.MACRO  INTOUTC2 D1,D2,D3
.NARG   CNT
.IF NB  D3
PUSHL   D3
.ENDC
.IF NB  D2
PUSHL   D2
.ENDC
.IF NB  D1
PUSHL   D1
.ENDC
.ENDM

.MACRO  INT_LW_CALL C
.IF EQ  ERR
BSBW    MAC$INTOUT_'C'_LW
.IFF
BSBW    MAC$INTERR_'C'_LW
.ENDC
.ENDM

.MACRO  $INTOUT_WD      ACTN,DATA
MOVZBL  #ACTN,R0
MOVZWL  DATA,R1
BSBW    MAC$INTOUT_WD
.ENDM
```

```
        .SBTTL  PUSH/POP ON VALUE STACK

;++
;       THESE TWO MACROS ARE USED TO PUSH AND POP FROM THE
;       PARSER VALUE STACK.
;--

        .MACRO  $VPUSH  LOC
        INCL    R7
        MOVL    LOC,W^MAC$AL_VALSTACK[R7]
        .ENDM   $VPUSH

        .MACRO  $VPOP   LOC
        MOVL    W^MAC$AL_VALSTACK[R7],LOC
        DECL    R7
        .ENDM   $VPOP

        .SBTTL  ADJUST PC

        .MACRO  $INC_PC CNT
        .IF     B       CNT
        INCL    W^MAC$GL_PC
        .IFF
        ADDL2   CNT,W^MAC$GL_PC
        .ENDC
        .ENDM   $INC_PC

        .MACRO  $DEC_PC CNT
        .IF     B       CNT
        DECL    W^MAC$GL_PC
        .IFF
        SUBL2   CNT,W^MAC$GL_PC
        .ENDC
        .ENDM

        .SBTTL  OUTPUT BYTE TO OBJECT CODE

        .MACRO  $OBJ_OUTBYT     BYT
        .IF DIF <BYT>,<R0>
        MOVB    BYT,R0
        .ENDC
        BSBW    MAC$OUTOBJ
        .ENDM   $OBJ_OUTBYT

        .MACRO  $OBJ_OUTBYT_0   BYT
        MOVB    BYT,(R10)+
        .ENDM   $OBJ_OUTBYT_0


.SBTTL  OUTPUT BYTE TO OBJECT CODE, CHECKING FOR BUFFER OVERFLOW

.MACRO  $OBJ_CHKBYT     BYT
.IF DIF <BYT>,<R0>
MOVB    BYT,R0
.ENDC
BSBW    MAC$CHKBYT
.ENDM   $OBJ_CHKBYT

.SBTTL  SIGNAL PASS 2 ERROR

.MACRO  $MAC_P2_ERR     MSGNAM
MOVZWL  #<MAC$_'MSGNAM&^XFFFF>,R0
BSBW    MAC$PASS_2_ERR
```

```
.ENDM   $MAC_P2_ERR

.SBTTL  OUTPUT BYTES TO LISTING FILE

.MACRO  $MAC_LIST_BYTE  N
.IF DIF <N>,<R0>
MOVZBL  N,R0
.ENDC
BSBW    MAC$LIST_BYTES
.ENDM   $MAC_LIST_BYTE

        .SBTTL  SET MACRO ERROR CODE

        .MACRO  $MAC_ERR CODE
        MOVZWL  #<MAC$_'CODE&^XFFFF>,R0
        .ENDM   $MAC_ERR
```

```
        .SBTTL  $MAC_INSERT_SYM MACRO TO PLACE SYMBOLS IN LINKED LIST
;++
;       $MAC_INSERT_SYM -- PLACE SYMBOLS IN ONE LINKED LIST
;
; MACRO TO PLACE SYMBOLS IN ONE LINKED LIST.  THIS MACRO IS
; USED WHEN ONLY A SINGLE LINKED LIST INSTEAD OF A HASH TABLE
; IS USED.  THE SYMBOL 'INSYMP' IS THE LINK POINTER, AND
; IT SHOULD BE ZEROED BEFORE THE LIST IS CREATED.
;
;--

        .MACRO  $MAC_INSERT_SYM NAM,VL1,FLAG=0,HEAD     ;NAME, VALUE, FLAG, HEAD
        .ASCIC  /NAM7
        .IIF    NOT_BLANK <HEAD>, HEAD::
        INSYTM=.
        .LONG   INSYMP
        INSYMP = INSYTM
        .NCHR   INSYMC,<NAM>
        .BYTE   INSYMC+1
        .LONG   VL1                     ;VALUE
        .WORD FLAG                      ;FLAGS
        .BYTE   0                       ;TOKEN
        .BYTE   0                       ;SEGMENT
        .ENDM   $MAC_INSERT_SYM

        .MACRO  $MAC_INSERT_SYX NAM,VAL,HEAD            ;NAME, VALUE, HEAD
        .ASCIC  /NAM7
        .IIF    NOT_BLANK <HEAD>, HEAD::
        INSYTM=.
        .LONG   INSYMP
        INSYMP = INSYTM
        .NCHR   INSYMC,<NAM>
        .BYTE   INSYMC+1
        .LONG   VAL
        .ENDM   $MAC_INSERT_SYX

        .LIST
        .END
```

DATA
LIS

DEFINE
LIS

FLOAT
LIS

ERRMSG
LIS

GETARG
LIS

DATA1
LIS

INPUT
LIS

ERROR
LIS

FINISH
LIS

GETCMD
LIS