

MMM		MMM	AAAAAAAAAA	CCCCCCCCCCCC	RRRRRRRRRRRR	0000000000		
MMM		MMM	AAAAAAAAAA	CCCCCCCCCCCC	RRRRRRRRRRRR	0000000000		
MMM		MMM	AAAAAAAAAA	CCCCCCCCCCCC	RRRRRRRRRRRR	0000000000		
MMMMMMM	MMMMMMM	AAA	AAA	CCC	RRR	RRR	000	000
MMMMMMM	MMMMMMM	AAA	AAA	CCC	RRR	RRR	000	000
MMMMMMM	MMMMMMM	AAA	AAA	CCC	RRR	RRR	000	000
MMM	MMM	MMM	AAA	AAA	CCC	RRR	RRR	000
MMM	MMM	MMM	AAA	AAA	CCC	RRR	RRR	000
MMM	MMM	MMM	AAA	AAA	CCC	RRR	RRR	000
MMM	MMM	MMM	AAA	AAA	CCC	RRR	RRR	000
MMM	MMM	MMM	AAA	AAA	CCC	RRR	RRR	000
MMM	MMM	MMM	AAA	AAA	CCC	RRR	RRR	000
MMM	MMM	MMM	AAA	AAA	CCC	RRR	RRR	000
MMM	MMM	MMM	AAA	AAA	CCC	RRR	RRR	000
MMM	MMM	MMM	AAAAAAAAAAAAAAAA	CCC	RRR	RRR	000	000
MMM	MMM	MMM	AAAAAAAAAAAAAAAA	CCC	RRR	RRR	000	000
MMM	MMM	MMM	AAAAAAAAAAAAAAAA	CCC	RRR	RRR	000	000
MMM	MMM	MMM	AAA	AAA	CCC	RRR	RRR	000
MMM	MMM	MMM	AAA	AAA	CCC	RRR	RRR	000
MMM	MMM	MMM	AAA	AAA	CCC	RRR	RRR	000
MMM	MMM	MMM	AAA	AAA	CCC	RRR	RRR	000
MMM	MMM	MMM	AAA	AAA	CCCCCCCCCCCC	RRR	RRR	0000000000
MMM	MMM	MMM	AAA	AAA	CCCCCCCCCCCC	RRR	RRR	0000000000
MMM	MMM	MMM	AAA	AAA	CCCCCCCCCCCC	RRR	RRR	0000000000

```

LL               IIIIII             SSSSSSSS
LL               IIIIII             SSSSSSSS
                II                  SS
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
LLLLLLLLLLLL    IIIIII             SSSSSSSS
LLLLLLLLLLLL    IIIIII             SSSSSSSS

```

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

.NLIST
 .TITLE MAC\$DEFINE
 .IDENT 'V04-000'

MACRO LIBRARY DEFINITION FILE FOR MACRO

 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY *
 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS. *
 * ALL RIGHTS RESERVED. *

* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED *
 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE *
 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER *
 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY *
 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY *
 * TRANSFERRED. *

* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE *
 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT *
 * CORPORATION. *

* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS *
 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL. *

++
 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 SYSS\$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 .ASCII strings. QAR #890 and
 SPR #11-42904.

ML=
 MF=
 MD=
 MQ=
 MG=
 MO=
 MH=
 WB=
 WW=
 WL=
 WF=
 WD=
 WQ=
 WG=
 WO=
 WH=
 VB=
 VW=
 VL=
 VF=
 VD=
 VQ=
 VG=
 VO=
 VH=

ERR
ERR
ERR
ERR
ERR
ERR
ERR
GOA
DEC
ID
MAC
DOP
DCQ
DCQ
DEQ
DPC
DSQ
DSQ
DAN
DAN
DOP
DCL
DPL
DMI
DTI
DDI
DAN
DOR
DXO
DAT
DPO
DIN
DUP
DUP
DUP
DUP
DUP
DUP
DBU
DGU
DIU
DLL
DSL
DWU
RRR
DMA
KAS
KAS
KAS
KPA

KAD
KBY
KDO
KFI
KFL
KLO
KQU
KWO
KBU
KBU
KBL
KBL
KBL
KBL
KBL
KIF
KII
KIF
KIF
KIF
KIR
KIR
KRE
KEN
KEN
KMA
KMC
KME
KEN
KMD
KDE
KDS
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
KER
KPR
KWA
KNT
KOP

.SBTTL DECLARATIONS

:
: INCLUDE FILES:
::
: MACROS:
::
: EQUATED SYMBOLS:
::
: OWN STORAGE:
:KENI
KXFE
KASO
KCRO
KNCF
KDFI
KSGN
KSGN
KBLI
KBLI
KBLI
KGFI
KHFI
KOCI
KREI
KLII

.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$S_LINK:      .BLKL  1          ;LINK TO NEXT SYMBOL OR 0
SYM$B_NAME:      .BLKB  1          ;OFFSET FROM BASE TO NAME
SYM$S_VAL:       .BLKL  1          ;SYMBOL VALUE (32 BITS)
SYM$W_FLAG:      .BLKW  1          ;SYMBOL FLAGS
SYM$B_TOKEN:     .BLKB  1          ;TOKEN VALUE
SYM$B_SEG:       .BLKB  1          ;SEGMENT (PSECT) DEFINED IN
SYM$K_BLK$IZ:    .BLKB  1          ;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\$S_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$S_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

```

```

$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$LINK: .BLKL 1 ;LINK TO NEXT PSECT NAME BLOCK OR 0
PSC$NAME: .BLKB 1 ;OFFSET FROM BASE TO PSECT NAME
PSC$MAXLGTH: .BLKL 1 ;PSECT MAXIMUM LENGTH
PSC$FLAG: .BLKW 1 ;PSECT FLAGS
PSC$UNUSED: .BLKB 1 ;UNUSED BYTE
PSC$SEG: .BLKB 1 ;PSECT SEGMENT NUMBER
PSC$OPTIONS: .BLKW 1 ;PSECT OPTIONS
PSC$CURLOC: .BLKL 1 ;PSECT CURRENT LOCATION
PSC$K_BLKSIZE: ;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_ALCOPTNS=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>
$PSC_BITDEF      SH
PSC$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 ;DEFAULTLT PSECT OPTIONS

.PSECT
.MDELETE          $MAC SYMBLKDEF, $SYM_BITDEF, $PSC_BITDEF
.ENDM             $MAC_SYMBLKDEF

```

CRF
CRF
CRF
CRF
CRF
CRF
CRF
CRF
CRF

```
.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 \$ABSS,ABS

 $\cdot = 0$

INP\$LINK:	.BLKL	1	;LINK TO NEXT BLOCK
INP\$NXTL:	.BLKL	1	;LINK TO NEXT LINE
INP\$GETL:	.BLKL	1	;ADDRESS OF ROUTINE TO GET NEXT LINE
INP\$IFLVL:	.BLKL	1	; "IF" LEVEL COUNTER
INP\$IFVAL:	.BLKL	1	; "IF" VALUE
INP\$RPTCNT:	.BLKL	1	;COUNTER FOR REPEAT/IRP
INP\$PAGP:	.BLKL	1	;POINTER TO PAGES USED BY THIS INPUT
			;BLOCK
INP\$B_ARGCT:	.BLKB	1	;ARGUMENT COUNT
INP\$ARGS:	.BLKL	1	;POINTERS TO ARGUMENTS
INP\$K_BLKSIZ:			;SIZE OF AN INPUT BLOCK

 $\rho = 0$

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

```
.MDELETE          SMAC_INPBLKDEF
.ENDM          SMAC_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 \$ABSS,ABS

 $\rho = 0$

NAME	DATA TYPE	DESCRIPTION
MNB\$LINK:	.BLKL 1	: LINK TO NEXT MNB
MNB\$B_NAME:	.BLKB 1	: OFFSET FROM BASE TO MACRO NAME
MNB\$TXTP:	.BLKL 1	: POINTER TO TEXT
MNB\$W_FLAG:	.BLKW 1	: FLAGS
MNB\$PAGP:	.BLKL 1	: POINTER TO PAGES USED TO DEFINE MACRO
MNB\$PAGC:	.BLKL 1	: COUNT OF PAGES USED
MNB\$CRSYMF:	.BLKL 1	: CREATED SYMBOL FLAGS
MNB\$B_ARGCT:	.BLKB 1	: ARGUMENT COUNT (255. MAX)
MNB\$ARGP:	.BLKL 1	: POINTER TO ARG LIST
MNB\$K_BLKSI:	.BLKL 1	: 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 = ^015 ;CARRIAGE RETURN
TAB = ^011 ;HORIZONTAL TAB
FF = ^014 ;FORM FEED
BLNK = ^040 ;BLANK
HYPHEN = ^A/-/ ;HYPHEN
SEMI = ^A/;/ ;SEMI COLON
CHRSV_CVTLWC = ^A/a/ ;CONVERT ANYTHING .GT. LOWERCASE A
CHRSV_NOCVT = ^X7F ;DO NOT CONVERT TO UPPERCASE
INTSK_BUFSIZ = <10*512>-<3*4> ;SIZE OF INT. BUFFER
INTSK_BUFWRN = INTSK_BUFSIZ-100 ;WARNING LIMIT
INPSK_BUFSIZ = 1000. ;Input buffer size
OBJSK_BUFSIZ = 512 ;OBJECT FILE BUFFER SIZE
LSTSK_BUFSIZ = 134. ;LISTING FILE BUFFER SIZE
LSTSK_TITLE_SIZ = 40. ;Size of title sub-string
REGS_PC = 15 ;PC REGISTER
RDXSV_BINARY = 0 ;INDEX FOR BINARY RADIX
RDXSV_OCTAL = 1 ;INDEX FOR OCTAL RADIX
RDXSV_DECIMAL = 2 ;INDEX FOR DECIMAL RADIX
RDXSV_HEX = 3 ;INDEX FOR HEX RADIX
RDXSV_FLOAT = 4 ;INDEX FOR F FLOATING NUMBERS
RDXSV_DOUBLE = 5 ;INDEX FOR D FLOATING NUMBERS
RDXSV_GFLOAT = 6 ;INDEX FOR G FLOATING NUMBERS
RDXSV_HFLOAT = 7 ;INDEX FOR H FLOATING NUMBERS
LSTSK_L_P_PAGE = 60 ;LINES PER PAGE IN LISTING
STBSK_PG_MISS = 10 ;# PAGES TO ALLOCATE FOR EMPTY SYMBOL BUCKET
SYMSK_MAXLEN = 31 ;Maximum characters/symbol
SYMSK_TWOCOL = 16 ;Character size for 2 column symbol table
CHRSK_SPA_MSK = ^X01 ;BIT MASK FOR SPACE-LIKE CHARACTER
CHRSV_SPA_MSK = 0 ;BIT NUMBER FOR SPACE-LIKE CHAR
CHRSK_SYM_DLM = ^X02 ;BIT MASK FOR SYMBOL DELIMITER CHARACTER
CHRSV_SYM_DLM = 1 ;BIT NUMBER FOR SYMBOL DELIM.
CHRSK_SYM_CHR = ^X04 ;BIT MASK FOR CHAR THAT CAN START SYMBOL
CHRSV_SYM_CHR = 2 ;BIT NUMBER FOR SYMBOL CHAR
CHRSK_SYM_CH1 = ^X08 ;BIT MASK FOR CHAR THAT CAN BE IN SYMBOL
CHRSV_SYM_CH1 = 3 ;BIT NUMBER FOR SYMBOL CHAR
CHRSK_NUM_BER = ^X10 ;BIT MASK FOR NUMBER
CHRSV_NUM_BER = 4 ;BIT NUMBER FOR NUMBER
CHRSK_COMMA_CR = ^X20 ;BIT MASK FOR COMMA AND CR
CHRSV_COMMA_CR = 5 ;BIT NUMBER FOR COMMA AND CR
CHRSK_ILL_CHR = ^X40 ;BIT MASK FOR ILLEGAL CHARACTER
CHRSV_ILL_CHR = 6 ;BIT NUMBER FOR ILLEGAL CHARACTER
ARGSK_SIZE = 1000 ;Maximum characters in MACRO argument
AUDSK_SIZE = 16 ;Audit trail string size

```

```

.MDELETE $MAC_GENVALDEF
.ENDM $MAC_GENVALDEF

```


++
[
--

\$MAC_MTXDEF

DEFINES THE MACRO TEXT SPECIAL MARKERS USED DURING MACRO
AND REPEAT/IRP PROCESSING.

.MACRO \$MAC_MTXDEF

MTXS_ARGMRK	=	^XFF	;ARG NO. FOLLOWS (BYTE)
MTXS_TXTLNK	=	^XFE	;LONGWORD OF LINK FOLLOWS
MTXS_LXLEN	=	^XFD	;LEXICAL OPERATOR--LENGTH
MTXS_LXEXT	=	^XFC	;LEXICAL OPERATOR--EXTRACT
MTXS_LXLOC	=	^XFB	;LEXICAL OPERATOR--LOCATE
:			
MTXS_LITSTR	=	^XFA TO ^XFO	RESERVED
MTXS_LITVAL	=	^XEF	;LITERAL STRING FOLLOWS
MTXS_SYMADR	=	^XEE	;LITERAL VALUE FOLLOWS
MTXS_NOMORE	=	^XED	;SYMBOL ADDRESS FOLLOWS
		^XEC	;NO MORE ARGUMENTS

.ENDM \$MAC_MTXDEF

++
: MTXS_ARGMRK FOLLOWED BY A BYTE CONTAINING THE ARGUMENT NUMBER
: MTXS_TXTLNK FOLLOWED BY A LONGWORD OF ADDRESS (TEXT LINK)
: MTXS_LITSTR FOLLOWED BY WORD OF COUNT, THEN ASCII STRING
: MTXS_LITVAL FOLLOWED BY LONGWORD OF VALUE
: MTXS_SYMADR FOLLOWED BY LONGWORD OF SYMBOL ADDRESS
--

++
[
--

:
:
:
S

:
:
:
E

:
:
:
T

```
.SBTTL  DEFINE MACRO LIBRARY FILE OFFSETS
.MACRO  $MAC_MLFDEF
$NAMDEF
.PSECT  $ABSS,ABS

.=0
    MLFSK_RSFNLN    = NAMSC_MAXRSS ; Resultant file name length
MLFSL_QLINK:        .BLKL  2        ;QUEUE LINK WORDS
MLFSL_MCDEF:        .BLKL  1        ;# MACROS DEFINED IN THIS MLB
                                ; (I.E. IMCALL'S SATISFIED HERE)
MLFSQ_FNAMDS:       .BLKQ  1        ; File name descriptor
MLFSL_CTINDEX:      .BLKL  1        ; Control table index
MLFSX_NAMBLK:       .BLKB  NAMSC_BLN ; NAM block
MLFST_FNAM:         .BLKB  MLFSK_RSFNLN ; File name string
MLFSK_BLKSIZ:

.MDELETE            $MAC_MLFDEF
.ENDM               $MAC_MLFDEF
```



```

:++
: 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$M_MODE = 1aOPD$S_MODE-1aOPD$V_MODE ; Mask to isolate mode value
OPD$M_ADDR      =      0a<OPD$V_MODE> ;ADDRESS ACCESS MODE
OPD$M_READ      =      1a<OPD$V_MODE> ;READ ACCESS MODE
OPD$M_MODIFY     =      2a<OPD$V_MODE> ;MODIFY ACCESS MODE
OPD$M_WRITE     =      3a<OPD$V_MODE> ;WRITE ACCESS MODE
OPD$M_VFIELD    =      4a<OPD$V_MODE> ;FIELD ACCESS
OPD$M_BB        =      5a<OPD$V_MODE>+1;BRANCH BYTE
OPD$M_BW        =      6a<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=OPDSM_MODIFY+L
MF=OPDSM_MODIFY+F
MD=OPDSM_MODIFY+D
MQ=OPDSM_MODIFY+Q
MG=OPDSM_MODIFY+G
MO=OPDSM_MODIFY+O
MH=OPDSM_MODIFY+H
WB=OPDSM_WRITE+B
WW=OPDSM_WRITE+W
WL=OPDSM_WRITE+L
WF=OPDSM_WRITE+F
WD=OPDSM_WRITE+D
WQ=OPDSM_WRITE+Q
WG=OPDSM_WRITE+G
WO=OPDSM_WRITE+O
WH=OPDSM_WRITE+H
VB=OPDSM_VIELD+B
VW=OPDSM_VIELD+W
VL=OPDSM_VIELD+L
VF=OPDSM_VIELD+F
VD=OPDSM_VIELD+D
VQ=OPDSM_VIELD+Q
VG=OPDSM_VIELD+G
VO=OPDSM_VIELD+O
VH=OPDSM_VIELD+H
```

```
.ENDM $MAC_OPRDEF
```

```
.SB
.MA
.IF
MOV
.EN
BSB
.EN
.SB
.MA
MOV
BSB
```


.SBTTL TERMINAL GRAMMAR SYMBOL DEFINITIONS

H 5

++
--

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.

.ENI
.SB
.MA
.IF
MOV
.ENI
BSB
.ENI

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

.SBTTL FLAG DEFINITIONS

K 5

MACS
V04

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
FLGSV 'F' = XT
FLGSM 'F' = X2
X1 = X1 + 1
X2 = X2 @ 1
.ENDM $FLG_DEFINE
```

X1=0
X2=1

\$FLG_DEFINE	ALLCHR	:TRUE INDICATES: :MAC\$GETCHR SHOULD PASS SEMICOLONS :(ALLCHR MUST BE LOW BIT!!!)
\$FLG_DEFINE	BOL	:OPCODE NOT YET FOUND
\$FLG_DEFINE	COMEXPR	:EXPRESSION IS COMPILE TIME
\$FLG_DEFINE	CONT	:ALLOW LINE CONTINUATIONS
\$FLG_DEFINE	DATRPT	:REPEATING DATA DEFINITION
\$FLG_DEFINE	ENDMCH	:CHECK MACRO DIRECTIVES
\$FLG_DEFINE	EVAEXPR	:EXPR TO BE EVAL ON PASS 2
\$FLG_DEFINE	EXOPT	: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 L 5
$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

MACS
Symb

MACS

PSEC

: A
: BL

Phas

Init
Comm
Pass
Symb
Pass
Symb
Psec
Cros
Asse

The
631
Ther
66 s
0 pa

Macr

\$25
\$25
TOTAL

0 GE

Ther

MACR

.SBTTL ADDRESSING MODE DEFINITIONS

N 5

***F1

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

.MACRO \$MAC_ADRMODDEF

X=0

.MACRO COD SYM

ADMS '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)	
	ADMS_MAXMOD	= X-1	:MAX. ALLOWABLE MODE

.MDELETE \$MAC_ADRMODDEF, COD

.ENDM \$MAC_ADRMODDEF

```

:++
: 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 ARITHMETIC 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 :EXCLUSIVE OR
;***ABOVE SYMBOLS ARE ARITHMETIC OPERATORS***POSITION MUST NOT CHANGE
INT ASN :ASSIGNMENT
INT AUGPC :AUGMENT PC

```



```

INT BDST      ;GENERATE BRANCH DESTINATION
INT CHKL      ;ALIGN 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

```

```

:++
:
: 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

SYMSF_WEAK	=	^0001	:	WEAK RESOLUTION
SYMSF_DEF	=	^0002	:	DEFINITION
SYMSF_UNI	=	^0004	:	UNIVERSAL
SYMSF_REL	=	^0010	:	RELATIVE
SYMSF_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 $INTOUT_LW      ACTN,DATA
INTOUTC2 DATA
MOVZBL #ACTN,R0
ERR=0
.IF IDN <ACTN>,<INT$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

.SBTL OUTPUT BYTES TO LISTING FILE

.MACRO $MAC_LIST_BYTE N
  .IF DIF <N>,ZRO>
  MOVZBL N,R0
  .ENDC
  BSBW $MAC_LIST_BYTES
  .ENDM $MAC_LIST_BYTE

.SBTL SET MACRO ERROR CODE

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


I 6

.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

0225 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY