


```

MM      MM      AAAAAA      CCCCCCCC      SSSSSSSS      UU      UU      BBBB8888
MM      MM      AAAAAA      CCCCCCCC      SSSSSSSS      UU      UU      88888888
MMMM    MMMM    AA      AA      CC      SS      UU      UU      BB      BB
MMMM    MMMM    AA      AA      CC      SS      UU      UU      BB      BB
MM      MM      AA      AA      CC      SS      UU      UU      BB      BB
MM      MM      AA      AA      CC      SS      UU      UU      BB      BB
MM      MM      AA      AA      CC      SSSSSS      UU      UU      88888888
MM      MM      AA      AA      CC      SSSSSS      UU      UU      88888888
MM      MM      AAAAAAAAAA      CC      SS      UU      UU      BB      BB
MM      MM      AAAAAAAAAA      CC      SS      UU      UU      BB      BB
MM      MM      AA      AA      CC      SS      UU      UU      BB      BB
MM      MM      AA      AA      CC      SS      UU      UU      BB      BB
MM      MM      AA      AA      CCCCCCCC      SSSSSSSS      UUUUUUUUUU      88888888
MM      MM      AA      AA      CCCCCCCC      SSSSSSSS      UUUUUUUUUU      88888888
                                     ....
                                     ....
                                     ....
                                     ....

LL      I11111      SSSSSSSS
LL      I11111      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
LL      I11111      SSSSSSSS
LL      I11111      SSSSSSSS
  
```


(2)	59	DECLARATIONS
(5)	92	MAC\$FAOUT FORMAT ASCII STRINGS
(6)	134	MAC\$SET_PC RECORD HIGH WATER PC
(7)	170	MAC\$HASH_SYM FORM HASH VALUE FOR SYMBOL IN MAC\$AB_TMPSYM
(8)	200	TRUNCATION CHECK ROUTINES
(10)	327	MAC\$SRC_KEYS SEARCH KEYWORD LIST WITH ABBREVIATIONS
(11)	389	CONVERT_LOWER CASE TO UPPER CASE
(12)	407	MAC\$OPEN_INPUT OPEN INPUT FILE
(13)	515	MAC\$RESCANCH RESCAN CURRENT CHARACTER
(14)	532	MAC\$OPTIMIZEEXPR DELETE EXPRESSION
(15)	577	MAC\$SKP_OPR SKIP TO NEXT OPERAND OR EOL
(16)	613	MAC\$CLOSE_FILES CLOSE ALL FILES
(17)	656	MAC\$CLOSE_LIB CLOSE MACRO LIBRARIES
(18)	677	ALLOCATE/DEALLOCATE VIRTUAL MEMORY

```
0000 1      .TITLE  MAC$MACSUB SUBROUTINES FOR VAX-11/780 ASSEMBLER
0000 2      .IDENT  'V04-000'
0000 3
0000 4
0000 5      *****
0000 6      *
0000 7      *  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 8      *  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 9      *  ALL RIGHTS RESERVED.
0000 10     *
0000 11     *  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 12     *  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 13     *  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 14     *  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 15     *  OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 16     *  TRANSFERRED.
0000 17     *
0000 18     *  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 19     *  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 20     *  CORPORATION.
0000 21     *
0000 22     *  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 23     *  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 24     *
0000 25     *
0000 26     *****
0000 27
0000 28
0000 29     ++
0000 30     FACILITY:      VAX MACRO ASSEMBLER OBJECT LIBRARY
0000 31
0000 32     ABSTRACT:
0000 33
0000 34     The VAX-11 MACRO assembler translates MACRO-32 source code into object
0000 35     modules for input to the VAX-11 LINKER.
0000 36
0000 37     ENVIRONMENT:  USER MODE
0000 38
0000 39     AUTHOR: Benn Schreiber, CREATION DATE: 21-AUG-78
0000 40
0000 41     MODIFIED BY:
0000 42
0000 43     V03-002      RRB0031      Rowland R. Bradley      09-Jul-1984
0000 44     Use Lib$Trim_Filespec to insert the filename on the
0000 45     listing.
0000 46
0000 47     V03-001      RRB0030      Rowland R. Bradley      08-Jul-1984
0000 48     Copy expanded filename string into the header. Expanded
0000 49     filename string is pointed to by FAB$FNA because the
0000 50     routine LIB$FIND_FILE was used to determine the name in
0000 51     module MAC$GETCMD. The name was subsequently copied from
0000 52     the result of the LIB$FIND_FILE.
0000 53
0000 54     V02-019      BLS0057      Benn Schreiber      13-Jun-1981
0000 55     Correct reference to SUM$INIT_EDIT to be General addressing mode
0000 56
0000 57     --
```



```
0000 59          .SBTTL  DECLARATIONS
0000 60          :
0000 61          : INCLUDE FILES:
0000 62          :
0000 63          :
0000 64          :
0000 65          : MACROS:
0000 66          :
0000 67          :
0000 68          $RABDEF          ;DEFINE RAB OFFSETS
0000 69          $FABDEF          ;DEFINE FAB OFFSETS
0000 70          $MAC_CTLFLGDEF    ;DEFINE CONTROL FLAGS
0000 71          $MAC_GENVALDEF    ;DEFINE GENERAL VALUES
0000 72          $MAC_MLFDEF      ; Define MLF offsets
0177 73          $MAC_SYMBLKDEF    ;DEFINE SYMBOL BLOCK OFFSETS
0000 74          $MACMSGDEF        ; Define message codes
0000 75          $MAC_MNBDEF        ; Define MXB offsets
0008 76          :
0008 77          :
0008 78          : EQUATED SYMBOLS:
0008 79          :
0008 80          :
0008 81          :
0008 82          : OWN STORAGE:
0008 83          :
0008 84          :
00000000 85          .PSECT  MAC$TEMP_STOR,WRT,GBL,LONG
0000 86  $DEF  MAC$AB_TMP_SPEC .BLKB 255 ; TMP STOR FOR LIB$TRIM_FILESPEC
00FF 87          .ALIGN  LONG
```

MAC\$MACSUB
V04-000

E 11
SUBROUTINES FOR VAX-11/780 ASSEMBLER
DECLARATIONS

16-SEP-1984 02:09:11 VAX/VMS Macro V04-00
5-SEP-1984 01:49:14 [MACRO.SRC]MACSUB.MAR;1

Page 3
(4)

0100 89
00000000 90

.PSECT MAC\$RO_CODE_P15,NOWRT,GBL, LONG


```
0000 92      .SBTTL MAC$FAOUT FORMAT ASCII STRINGS
0000 93
0000 94      :++
0000 95      : FUNCTIONAL DESCRIPTION:
0000 96      :
0000 97      : THESE THREE ROUTINES ARE USED TO FORMAT ASCII STRINGS
0000 98      : USING $FAOL. THERE ARE THREE ENTRY POINTS:
0000 99      :
0000 100     : MAC$FAOUTS      THE ARGUMENT LIST IS PUSHED ONTO THE STACK
0000 101     : BEFORE CALLING. THE CALLER MUST CLEAR THE
0000 102     : STACK ON RETURN
0000 103     :
0000 104     : MAC$WRT_FAOUTS  SAME AS MAC$FAOUTS EXCEPT THAT MAC$WRTLST
0000 105     : IS CALLED TO WRITE THE LINE BEFORE RETURNING.
0000 106     :
0000 107     : MAC$FAOUT      THE ARGUMENT LIST IS POINTED TO BY R1
0000 108     :
0000 109     : INPUTS:
0000 110     :
0000 111     : R0              FAO CONTROL STRING
0000 112     :
0000 113     : OUTPUTS:
0000 114     :
0000 115     : MAC$GL_LINELN  ADJUSTED AND READY TO CALL MAC$WRTLST
0000 116     :
0000 117     : --
0000 118     :
0000 119     MAC$WRT_FAOUTS::
51 04 AE 9E 0000 120     MOVAB 4(SP),R1          ;POINT TO THE ARGUMENT LIST
0004 121     BSBB MAC$FAOUT          ;FORMAT THE STRING
0006 122     BRW MAC$WRTLST          ;WRITE LINE TO LISTING AND RETURN
0009 123
0009 124     MAC$FAOUTS::
51 04 AE 9E 0009 125     MOVAB 4(SP),R1          ;POINT TO THE ARGUMENT LIST
000D 126     MAC$FAOUT::
000D 127     $FAOL_S CTRSTR=(R0),-          ;FORMAT THE STRING
000D 128     OUTBUF=L^MAC$G_LSTBUFDES,-
000D 129     OUTLEN=L^MAC$G_LINELN,-
000D 130     PRMLST=(R1)
0000'CF 00000000'8F C2 0024 131     SUBL2 #MAC$K_LIST_SIZE,W^MAC$GL_LINELN
05 002D 132     RSB
```

```
002E 134 .SBTTL MAC$SET_PC RECORD HIGH WATER PC
002E 135
002E 136 :++
002E 137 : FUNCTIONAL DESCRIPTION:
002E 138 :
002E 139 : THIS ROUTINE IS CALLED TO STORE THE PC IN PSC$L_MAXLGTH
002E 140 : FOR THE CURRENT PSECT IF IT IS HIGHER THAN THE STORED
002E 141 : PC THERE. IT SHOULD BE CALLED ANY TIME THE PC IS ADJUSTED
002E 142 : BACKWARDS, OR BEFORE CHANGING PSECTS.
002E 143 :
002E 144 :--
002E 145
002E 146 MAC$SET_PC::
002E 147 :
002E 147 MOVL W^MAC$GL_PSECTPTR,R0 ;POINT TO CURRENT PSECT
05 A0 0000'CF D1 0033 148 CMPL W^MAC$GL_PC,PSC$L_MAXLGTH(R0) ;CURRENT PC HIGHER THAN STORED?
05 A0 0000'CF 1F 0039 149 BLSSU 10$ ;IF LSSU NO
05 A0 0000'CF D0 003B 150 MOVL W^MAC$GL_PC,PSC$L_MAXLGTH(R0) ;YES--SET NEW PC
05 0041 151 10$: RSB
0042 152
0042 153 :++
0042 154 : FUNCTIONAL DESCRIPTION:
0042 155 :
0042 156 : THIS ROUTINE UPDATES MAC$GL_LSB TO THE NEXT AVAILABLE LSB.
0042 157 :
0042 158 :--
0042 159
0042 160 MAC$SET_NEW_LSB::
0042 161 :
0042 161 MOVAB W^MAC$GL_LSB,R0 ;POINT TO THE WORD
04 A0 60 D1 0047 162 CMPL (R0),4(R0) ;CAN WE JUST INCREMENT LSB?
04 A0 07 1F 004B 163 BLSSU 10$ ;IF LSSU NO--MUST USE HIGHEST LSB PLUS 1
04 A0 60 D0 004D 164 MOVL (R0),4(R0) ;YES--STORE HIGHEST LSB SO FAR
04 A0 60 D6 0051 165 INCL (R0) ;INCREMENT TO NEXT LSB
05 0053 166 RSB ;DONE
60 04 A0 01 C1 0054 167 10$: ADDL3 #1,4(R0),(R0) ;USE HIGHEST LSB + 1
05 0059 168 RSB
```


005A 170 .SBTTL MAC\$HASH_SYM FORM HASH VALUE FOR SYMBOL IN MAC\$AB_TMPSYM

005A 171

005A 172 :++

005A 173 : FUNCTIONAL DESCRIPTION:

005A 174 :

005A 175 : THIS ROUTINE ACCUMULATES THE HASH VALUE FOR A SYMBOL.

005A 176 :

005A 177 : INPUTS:

005A 178 :

005A 179 : MAC\$AB_TMPSYM THE SYMBOL NAME (LENGTH, NAME)

005A 180 :

005A 181 : OUTPUTS:

005A 182 :

005A 183 : MAC\$GL_HSHVAL ACCUMULATED HASH VALUE

005A 184 :

005A 185 :--

005A 186

005A 187 MAC\$HASH_SYM::

50	0000'CF	OF	BB	005A	188	PUSHR	#^M<R0,R1,R2,R3>
	51	80	9E	005C	189	MOVAB	W^MAC\$AB_TMPSYM,R0
	52	51	9A	0061	190	MOVZBL	(R0)+,R1
	53	80	D0	0064	191	MOVL	R1,R2
	51	53	9A	0067	192	MOVZBL	(R0)+,R3
	F7	52	C0	006A	193	ADDL2	R3,R1
0000'CF	51	FFFFF80	F5	006D	194	SOBGR	R2,10\$
		8F	CB	0070	195	BICL3	#^C<HASHSZ>,R1,-
				007A	196		W^MAC\$GL_HSHVAL
		OF	BA	007A	197	POPR	#^M<R0,RT,R2,R3>
			05	007C	198	RSB	

;SAVE REGISTERS
;POINT TO THE SYMBOL NAME BLOCK
;GET THE # OF CHARS IN NAME
;SET LOOP COUNT
;GET NEXT CHARACTER
;IMPROVE HASH VALUE
;LOOP FOR ALL CHARACTERS
;TRIM TO HASH TABLE SIZE
;AND STORE HASH VALUE
;RESTORE REGISTERS

```
007D 200 .SBTTL TRUNCATION CHECK ROUTINES
007D 201
007D 202 :++
007D 203 : FUNCTIONAL DESCRIPTION:
007D 204 :
007D 205 : THIS ROUTINE CHECKS THE VALUE ON THE TOP OF THE VALUE STACK
007D 206 : FOR UNSIGNED BYTE TRUNCATION.
007D 207 :
007D 208 : INPUTS:
007D 209 :
007D 210 : R5 POINTS TO VALUE TO CHECK
007D 211 :--
007D 212
007D 213 MAC$CK_BYT_TRU1::
55 04 AE DE 007D 214 MOVAL 4(SP),R5 ;POINT R5 TO THE WORD
06 0A 10 0081 215 BSBB MAC$CK_BYT_TRUN ;CHECK FOR TRUNCATION
02 50 E8 0083 216 BLBS R0,10$ ;BRANCH IF NO ERROR
01 A5 B4 0086 217 CLRW 2(R5) ;YES--TRIM TO A BYTE
01 A5 94 0089 218 CLRB 1(R5) ;
008C 219 10$: RSB ;DONE
008D 220 MAC$CK_BYT_TRUN::
02 A5 B5 008D 221 TSTW 2(R5) ;POSITIVE VALUE?
0C 12 0090 222 BNEQ 10$ ;IF NEQ NO
01 A5 B5 0092 223 TSTW 1(R5) ;YES--UPPER 3 BYTES MUST BE 0
38 12 0095 224 BNEQ MAC$TRUNC_ERR ;IF NEQ TRUNCATION ERROR
03 A5 95 0097 225 TSTB 3(R5) ;
33 12 009A 226 BNEQ MAC$TRUNC_ERR ;
0F 11 009C 227 BRB 20$ ;ALL IS WELL
FFFF 8F 02 A5 B1 009E 228 10$: CMPW 2(R5),#-1 ;UPPER WORD MUST BE ALL ONES
29 12 00A4 229 BNEQ MAC$TRUNC_ERR ;IF NEQ THEN TRUNCATION ERROR
FF 8F 01 A5 91 00A6 230 CMPB 1(R5),#-1 ;SECOND BYTE MUST BE ALL ONES
22 12 00AB 231 BNEQ MAC$TRUNC_ERR ;IF NEQ TRUNCATION ERROR
50 01 D0 00AD 232 20$: MOVL #1,R0 ;RETURN SUCCESS
05 00B0 233 RSB
00B1 234
00B1 235 :++
00B1 236 : FUNCTIONAL DESCRIPTION:
00B1 237 :
00B1 238 : THIS ROUTINE CHECKS THE VALUE ON THE TOP OF THE VALUE STACK
00B1 239 : FOR WORD TRUNCATION.
00B1 240 :
00B1 241 :--
00B1 242
00B1 243 MAC$CK_WRD_TRU1::
55 04 AE DE 00B1 244 MOVAL 4(SP),R5 ;POINT TO WORD IN QUESTION
07 10 00B5 245 BSBB MAC$CK_WRD_TRUN ;GO CHECK FOR TRUNCATION
03 50 E8 00B7 246 BLBS R0,10$ ;BRANCH IF ALL OK
02 A5 B4 00BA 247 CLRW 2(R5) ;NO--CLEAR TRUNCATION ERROR
05 00BD 248 10$: RSB ;ALL DONE
00BE 249 MAC$CK_WRD_TRUN::
02 A5 B5 00BE 250 TSTW 2(R5) ;UPPER WORD 0?
08 13 00C1 251 BEQL 10$ ;IF EQL YES--OK
FFFF 8F 02 A5 B1 00C3 252 CMPW 2(R5),#-1 ;NO--IS IT ALL ONES?
04 12 00C9 253 BNEQ MAC$TRUNC_ERR ;IF NEQ NO--TRUNCATION ERROR
50 01 D0 00CB 254 10$: MOVL #1,R0 ;RETURN SUCCESS
05 00CE 255 RSB
00CF 256
```



```

00CF 257 :
00CF 258 : REPORT TRUNCATION ERROR
00CF 259 :
00CF 260 :
00CF 261 MAC$TRUNC ERR:
09 6B 0E E1 00CF 262 BBC #FLG$V P2,(R11),10$ ;BRANCH IF PASS 1
00D3 263 $MAC_P2_ERR DATATRUNC ; Report error
05 00DB 264 RSB
00DC 265 10$: $MAC_ERR DATATRUNC ; Get message code
FF1C' 30 00E1 266 BSBW MAC$ERRORPT ;ISSUE ERROR TO PASS 2 AND RETURN
50 D4 00E4 267 CLRL R0 ;RETURN ERROR
05 00E6 268 RSB

```

```
00E7 270 :++
00E7 271 : FUNCTIONAL DESCRIPTION:
00E7 272 :
00E7 273 : CHECK FOR SIGNED BYTE TRUNCATION
00E7 274 :
00E7 275 :--
00E7 276 :
00E7 277 MAC$CK_SBY TRU1::
55 04 AE DE 00E7 278 MOVAL 4(SP),R5 ;POINT TO WORD IN QUESTION
10 14 10 00EB 279 BSBB MAC$CK_SBY_TRUN ;CHECK FOR TRUNCATION
50 50 E8 00ED 280 BLBS R0,20$ ;BRANCH IF ALL IS WELL
65 50 D4 00F0 281 CLRL R0 ;ASSUME POS BYTE
02 02 95 00F2 282 TSTB (R5) ;POS OR NEG. BYTE?
01 A5 50 18 00F4 283 BGEQ 10$ ;IF GEQ POS
01 A5 50 D7 00F6 284 DECL R0 ;NEGATIVE--MAKE -1
50 50 B0 00F8 285 10$: MOVW R0,2(R5) ;STORE -1
01 A5 50 90 00FC 286 MOVB R0,1(R5)
05 05 0100 287 20$: RSB ;DONE
0101 288 MAC$CK_SBY TRUN::
51 D4 0101 289 CLRL R1 ;ASSUME POSITIVE
65 95 0103 290 TSTB (R5) ;CHECK SIGN OF BYTE
02 18 0105 291 BGEQ 10$ ;BR IF GEQ
51 D7 0107 292 DECL R1 ;MAKE 0 INTO -1
02 A5 51 B1 0109 293 10$: CMPW R1,2(R5) ;DOES HIGH WORD HAVE RIGHT SIGN?
01 A5 C0 12 010D 294 BNEQ MAC$TRUNC_ERR ;IF NEQ NO--ERROR
51 91 010F 295 CMPB R1,1(R5) ;YES--HOW ABOUT SECOND BYTE?
50 BA 12 0113 296 BNEQ MAC$TRUNC_ERR ;IF NEQ NO
01 D0 0115 297 MOVL #1,R0 ;RETURN SUCCESS
05 05 0118 298 RSB
0119 299
0119 300 :++
0119 301 : FUNCTIONAL DESCRIPTION:
0119 302 :
0119 303 : CHECK FOR SIGNED WORD TRUNCATION
0119 304 :
0119 305 :--
0119 306 :
0119 307 MAC$CK_SWD TRU1::
55 04 AE DE 0119 308 MOVAL 4(SP),R5 ;POINT TO WORD IN QUESTION
10 10 10 011D 309 BSBB MAC$CK_SWD_TRUN ;CHECK FOR TRUNCATION
0C 50 E8 011F 310 BLBS R0,20$ ;BRANCH IF ALL IS WELL
50 50 D4 0122 311 CLRL R0 ;ASSUME POS. BYTE
65 50 B5 0124 312 TSTW (R5) ;POSITIVE?
02 02 18 0126 313 BGEQ 10$ ;IF GEQ YES
01 A5 50 D7 0128 314 DECL R0 ;NO--MAKE -1
50 50 B0 012A 315 10$: MOVW R0,2(R5) ;FIX THE ERROR
05 05 012E 316 20$: RSB
012F 317 MAC$CK_SWD TRUN::
51 D4 012F 318 CLRL R1 ;ASSUME POSITIVE
65 B5 0131 319 TSTW (R5) ;CHECK SIGN OF WORD
02 18 0133 320 BGEQ 10$ ;BR IF GEQ
51 D7 0135 321 DECL R1 ;MAKE 0 INTO -1
02 A5 51 B1 0137 322 10$: CMPW R1,2(R5) ;SIGN OF UPPER WORD CORRECT?
50 92 12 0138 323 BNEQ MAC$TRUNC_ERR ;IF NEQ NO--ERROR
01 D0 013D 324 MOVL #1,R0 ;RETURN SUCCESS
05 05 0140 325 RSB
```



```
0141 327 .SBTTL MAC$SRC_KEYS SEARCH KEYWORD LIST WITH ABBREVIATIONS
0141 328
0141 329 :++
0141 330 : FUNCTIONAL DESCRIPTION:
0141 331
0141 332 : THIS ROUTINE SEARCHES A LINKED LIST FOR A KEYWORD, WITH
0141 333 : ABBREVIATIONS TAKEN INTO ACCOUNT.
0141 334
0141 335 : INPUTS:
0141 336
0141 337 : R5 POINTER TO FIRST ELEMENT OF LINKED SYMBOL LIST
0141 338
0141 339 : OUTPUTS:
0141 340
0141 341 : R0 0 NOT FOUND OR MULTIPLE PARTIAL MATCHES
0141 342 : 1 FOUND
0141 343 : R1 0 NOT FOUND OR MULTIPLE PARTIAL MATCHES
0141 344 : <>0 ADDRESS OF SYMBOL BLOCK.
0141 345
0141 346 :--
0141 347
0141 348 MAC$SRC_KEYS::
0141 349 :
0141 350 : PUSH R4,R5,R6,R7,R8,R9 ;SAVE REGISTERS
0141 351 : MOV R5,R6 ;POINT TO LINKED LIST
0141 352 : MOVAB W^MAC$AB_TMSYM,R7 ;POINT TO SYMBOL IN QUESTION
0141 353 : MOVZBL (R7)+,R8 ;GET LENGTH OF SYMBOL
0141 354 : CLRL R9 ;CLEAR PARTIAL MATCH INDICATOR
0141 355 :
0141 356 : CHAIN THROUGH LIST LOOKING FOR SYMBOL
0141 357 :
0141 358 : 10$: MOVZBL SYMSB_NAME(R6),R0 ; Get offset to symbol count/name
0141 359 : SUBL3 R0,R6,R0 ; and form its address
0141 360 : MOVZBL (R0)+,R1 ; Get count byte and advance pointer
0141 361 : MATCHC R8,(R7),R1,(R0) ; Find the substring in here?
0141 362 : BEQL 60$ ;IF EQL YES
0141 363 : 20$: MOVL SYMSL_LINK(R6),R6 ; No--link to next
0141 364 : BNEQ 10$ ;IF NEQ GO CHECK IT OUT
0141 365 : 30$: CLRQ R0 ;ASSUME NOT FOUND
0141 366 : TSTL R9 ;WAS THERE A PARTIAL MATCH?
0141 367 : BEQL 50$ ;IF EQL NO--RETURN NOT FOUND
0141 368 : INCL R0 ;YES--RETURN 1 FOR FOUND
0141 369 : MOVL R9,R1 ;RETURN ADDRESS IN R1
0141 370 : BRB 50$ ;GO RETURN
0141 371 : 40$: MOVL #1,R0 ;RETURN SUCCESS
0141 372 : MOVL R6,R1
0141 373 : 50$: POPR #^M<R4,R5,R6,R7,R8,R9> ;RESTORE REGISTERS
0141 374 : RSB
0141 375 :
0141 376 : : HERE IF FOUND
0141 377 :
0141 378 : 60$: MOVZBL SYMSB_NAME(R6),R0 ; Get offset to symbol count/name
0141 379 : SUBL3 R0,R6,R0 ; and form its address
0141 380 : CMPB (R7),1(R0) ; Did it match on first character?
0141 381 : BNEQ 20$ ;IF NEQ NO--NOT A MATCH
0141 382 : TSTL R2 ;WAS IT A PERFECT MATCH?
0141 383 : BEQL 40$ ;IF EQL YES--GO RETURN
```

59	D5	0193	384	TSTL	R9	:NO--WAS THERE ALREADY A PARTIAL MATCH?
	D2	12	0195	BNEQ	30\$	:IF NEQ YES--CONFLICT MEANS NOT FOUND
	D0	0197	386	MOVL	R6, R9	:NO--SAVE ADDRESS OF THIS PARTIAL MATCH
	C8	11	019A	BRB	20\$	:CONTINUE LOOKING

[illegible]


```

019C 389 .SBTTL CONVERT LOWER CASE TO UPPER CASE
019C 390
019C 391 :++
019C 392 : FUNCTIONAL DESCRIPTION:
019C 393
019C 394 : THIS ROUTINE CONVERTS THE (POSSIBLE) LOWER CASE LETTER
019C 395 : IN R10 TO UPPER CASE.
019C 396 :
019C 397 :--
019C 398
019C 399 MAC$CVT_LOWER::
61 8F 5A 91 019C 400 CMPB R10,#^A/^A/ + ^X20 ;CAN CHARACTER BE LOWER CASE?
7A 8F 5A 91 01A0 401 BLSSU 10$ ;IF LSSU NO
5A 20 8A 01A2 402 CMPB R10,#^A/^Z/ + ^X20 ;STILL CHECKING FOR LOWER CASE
03 1A 01A6 403 BGTRU 10$ ;IF GTRU NOT LOWER CASE
05 01A8 404 BICB2 #^X20,R10 ;YES--CONVERT TO UPPER CASE
01AB 405 10$: RSB

```

[illegible]

```
01AC 407      .SBTTL MAC$OPEN_INPUT OPEN INPUT FILE
01AC 408
01AC 409      :++
01AC 410      : FUNCTIONAL DESCRIPTION:
01AC 411
01AC 412      : THIS ROUTINE IS CALLED TO OPEN THE INPUT FILE
01AC 413
01AC 414      : INPUTS:
01AC 415
01AC 416      : R0      ADDRESS OF FDB
01AC 417
01AC 418      : OUTPUTS:
01AC 419
01AC 420      : FILE IS OPENED.  CREATION TIME PLACED IN SUBTITLE BUFFER.
01AC 421
01AC 422      :--
01AC 423
01AC 424      MAC$OPEN_INPUT::
01AC 425      PUSHR    #^M<R1,R2,R3,R4,R5,R6,R7>
01B0 426          :SAVE REGISTERS
01B0 427      MOVL    R0,W^MAC$GL_CURINFDB      :SET AS CURRENT FDB
0000'CF 50  D0 01B0 427      MOVAB    8(R0),W^MAC$INPUT_RAB+RAB$FAB :POINT RAB TO FAB
003C'CF 08 A0 9E 01B5 428      MOVAB    #2,FAB$B_FSZ+8(R0)      :SET FIXED AREA SIZE OF 2
47 A0 02 90 01BB 429      MOVAB    FAB$B_RFM+8(R0)      :STACK RECORD FORMAT ADDRESS
27 A0 9F 01BF 430      PUSHAB    FAB=8(R0),-      :OPEN THE FILE
01C2 431      $OPEN    FAB=8(R0),-
01C2 432      ERR=W^MAC$ERR_OPN_INP
03 50 E8 01D0 433      BLBS    R0,5$      ; Branch if OK
00E0 31 01D3 434      BRW     100$
01D6 435      5$:
00 6B 19 E5 01D6 436      BBCC    #FLG$V_SEQFIL,(R11),10$ :ASSUME NOT SEQUENCED FILE
03 9E 91 01DA 437      CMPB    @($P)+,FAB$C_VFC      :IS IT SEQUENCED?
20$ 20$      BNEQ    20$      :IF NEQ NO
00 6B 19 E3 01DF 439      BBCC    #FLG$V_SEQFIL,(R11),20$ :YES--SET FLAG FOR PASS 2
50 0000'CF 9E 01E3 440      MOVAB    W^MAC$INPUT_RAB,R0      :POINT TO INPUT RAB
2C A0 0000'CF 9E 01E8 441      MOVAB    W^MAC$GL_RECHDBUF,RAB$RHB(R0) :SET RECORD HEADER BUFFER ADDRESS
01EE 442      $CONNECT RAB=(R0),-      :CONNECT RECORD STREAM
01EE 443      ERR=W^MAC$ERR_CONNECT
03 50 E8 01FB 444      BLBS    R0,21$
00B5 31 01FE 445      BRW     100$      :BRANCH IF ERROR
0201 446      21$:      $ASCTIM_S TIMBUF=L^MAC$AL_FTIM_DSC,- :CONVERT FILE CREATION DATE/TIME
0201 447      TIMADR=L^MAC$INPUT_XAB+XAB$Q_CDT ;
0218 448      :
0218 449      : COPY FILENAME AND CREATION DATE TIME TO VM FOR PASS 2 (IF PASS 1)
0218 450
0218 451      :
0218 452      : COPY FILENAME INTO SUBTITLE BUFFER
0218 453      : FIRST MAKE SURE THE FILENAME FITS IN THE SPACE PROVIDED
0218 454
0218 455      :
0218 456      :
0218 457      : CREATE TWO DESCS
0218 458      : POINT ONE TO THE CURRENT FILE SPEC
0218 459      : POINT OTHER TO SUBTITLE FILE SPEC
0218 460      : THEN TRIM TO FIT
0218 461
0218 462
50 00000000'EF D0 0218 463      MOVL    MAC$GL_CURINFDB,R0      :POINT TO FDB AGAIN
```



```

      51 3C A0 9A 021F 464      MOVZBL FAB$B_FNS+8(R0),R1      ;GET LENGTH OF FILENAME STRING
      52 34 A0 D0 0223 465      MOVL   FAB$B_FNA+8(R0),R2      ;GET ADDRESS OF FILENAME STRING
00000000'EF 62 51 28 0227 466      MOV C3  R1,(R2),-
                                L^MAC$AB_TMP_SPEC
                                -(SP)
      0000'CE 00'8F 7E 7C 022F 468      CLRQ   -(SP)
      0000'CE 00'8F 90 0231 469      MOV B   #DSC$K_CLASS,S,-      ;DESC CLASS
                                DSC$B_CLASS(SP)
      0000'CE 00'8F 90 0237 470      MOV B   #DSC$K_DTYPE,T,-      ;DESC TYPE
                                DSC$B_DTYPE(SP)
50 00000000'EF D0 023D 473      MOVL   MAC$GC_CURINFDB,R0      ;POINT TO FDB AGAIN
      51 3C A0 9A 0244 474      MOVZBL FAB$B_FNS+8(R0),R1      ;AGAIN...GET LENGTH OF FILENAME STRING
      0000'CE 51 90 0248 475      MOV B   R1,DSC$W_LENGTH(SP)      ;GET CURRENT LENGTH
0000'CE 00000000'EF DE 024D 476      MOVAL  L^MAC$AB_TMP_SPEC,-
                                DSC$A_POINTER(SP)
                                ;GET ADDRESS OF RESULT
      57 5E D0 0256 478      MOVL   SP,R7      ;SETUP POINTER TO STR DESC.
                                479
      0000'CE 00'8F 7E 7C 0259 480      CLRQ   -(SP)
      0000'CE 00'8F 90 025B 481      MOV B   #DSC$K_CLASS,S,-      ;DESC CLASS
                                DSC$B_CLASS(SP)
      0000'CE 00'8F 90 0261 482      MOV B   #DSC$K_DTYPE,T,-      ;DESC TYPE
                                DSC$B_DTYPE(SP)
      0000'CE 20 90 0267 484      MOV B   #32,DSC$W_LENGTH(SP)      ;GET CURRENT LENGTH
0000'CE 0000'CF DE 026C 486      MOVAL  W^MAC$AB_SBT_FILE,-
                                DSC$A_POINTER(SP)
                                ;GET ADDRESS OF RESULT
      56 5E D0 0273 488      MOVL   SP,R6      ;SETUP POINTER TO STR DESC.
                                489
      56 DD 0276 490      PUSHL  R6      ; OUTPUT STRING DESC ADDR
      57 DD 0278 491      PUSHL  R7      ; INPUT STRING DESC ADDR
00000000'GF 02 FB 027A 492      CALLS  #2,G^LIB$TRIM_FILESPEC      ; TRIM TO FIT
      32 50 E9 0281 493      BLBC   R0,100$      ; EXIT ON ERROR
                                494
50 00000000'EF D0 0284 495      MOVL   MAC$GL_CURINFDB,R0      ; Point to FDB again
      00 6B 27 E5 028B 496      BBCC   #FLG$V_UPDFIL,(R11),30$      ; Assume file is not being updated
                                497 30$:
      0000'CF 9F 028F 498      PUSHAB W^MAC$INPUT_RAB      ; Push parameters: RAB address
      00B8 C0 DD 0293 499      PUSHL  FAB$C_BLN+NAM$C_BLN+8(R0)      ; Update files list address
                                500
      00 6B 04 13 0297 501      BEQL   40$      ; If EQL zero their are no update files
      00 6B 27 E2 0299 502      BBSS   #FLG$V_UPDFIL,(R11),40$      ; Flag this file as being updated
                                503 40$:
      0000'CF 9F 029D 504      PUSHAB W^MAC$GT_SCB      ; SUM control block
00000000'GF 03 FB 02A1 505      CALLS  #3,G^SUM$INIT_EDIT      ; Initialise update files
      0B 50 E9 02A8 506      BLBC   R0,100$      ; Error if LBC
      5E 10 C0 02AB 507      ADDL   #16,SP      ;RESTORE STACK POINTER
      00FE 8F BA 02AE 508      POPR   #^M<R1,R2,R3,R4,R5,R6,R7>
                                ;RESTORE REGISTERS
      50 01 D0 02B2 509      MOVL   #1,R0      ;SET SUCCESS
      05 02B5 511      RSB
      02B6 512
      FD47' 31 02B6 513 100$: BRW   MAC$LAST_CHANCE      ;GO TO LAST CHANCE HANDLER
```

```
02B9 515      .SBTTL MAC$RESCANCH RESCAN CURRENT CHARACTER
02B9 516
02B9 517      ;++
02B9 518      ; FUNCTIONAL DESCRIPTION:
02B9 519      ;
02B9 520      ; THIS ROUTINE BACKS UP THE LINE POINTER AND RESETS THE NEXT
02B9 521      ; CHARACTER SO AS TO RESCAN THE CURRENT CHARACTER.
02B9 522      ;
02B9 523      ;--
02B9 524
02B9 525 MAC$RESCANCH::
50 0000'CF 9E 02B9 526      MOVAB W^MAC$GL_LINEPT,R0      ;GET POINTER TO MAC$GL_LINEPT
0000'8F 60 B1 02BE 527      CMPW (R0),#MAC$AB_LINEBF      ;AT BEGINNING OF LINE?
02 13 02C3 528      BEQL 10$      ;IF EOL YES
60 D7 02C5 529      DECL (R0)      ;NO--BACK IT UP
05 02C7 530 10$:      RSB
```



```
02C8 532 .SBTTL MAC$OPTIMIZEXPR DELETE EXPRESSION
02C8 533
02C8 534 :++
02C8 535 : FUNCTIONAL DESCRIPTION:
02C8 536 :
02C8 537 : MAC$OPTIMIZEXPR IS CALLED TO REMOVE THE CODE TO EVALUATE AN
02C8 538 : EXPRESSION FROM THE INTERMEDIATE BUFFER. THE EXPRESSION IS
02C8 539 : POINTED TO BY 'MAC$GL_EXPPTR' AND 'MAC$GL_EXPEND'. ANY MACRO
02C8 540 : TEXT WITHIN THIS RANGE IS COPIED DOWN. THE REST OF THE
02C8 541 : INTERMEDIATE CODE IS DELETED. ALL POINTERS AND COUNTERS
02C8 542 : ARE UPDATED (EXCEPT MAC$GL_EXPPTR AND MAC$GL_EXPEND).
02C8 543 :
02C8 544 :--
02C8 545
02C8 546 MAC$OPTIMIZEXPR::
58 01F8 8F BB 02C8 547 PUSHR #^M<R3,R4,R5,R6,R7,R8> ;PRESERVE REGISTERS
57 0000'CF D0 02CC 548 MOVL W^MAC$GL_EXPPTR,R8 ;POINT TO EXPRESSION START
57 0000'CF D0 02D1 549 MOVL W^MAC$GL_EXPEND,R7 ;AND END OF EXPRESSION
58 57 D1 02D6 550 CMPL R7,R8 ;IS THERE AN EXPRESSION?
58 3C 13 02D9 551 BEQL 50$ ;IF EQL NO
56 58 D0 02DB 552 MOVL R8,R6 ;COPY START OF EXPRESSION POINTER
FF 8F 66 91 02DE 553 10$: CMPB (R6),#^XFF ;MACRO LINE?
56 OD 13 02E2 554 BEQL 20$ ;IF EQL YES
50 66 9A 02E4 555 MOVZBL (R6),R0 ;NO--EXPRESSION COMMAND--GET LENGTH
56 50 C0 02E7 556 ADDL2 R0,R6 ;SKIP THE EXPRESSION
0000'CF 50 C0 02EA 557 ADDL2 R0,W^MAC$GL_INTCNT ;INCREASE REMAINING BYTES IN BUFFER
11 11 02EF 558 BRB 30$ ;
02F1 559 ;
02F1 560 : MACRO LINE--COPY DOWN
02F1 561 :
50 02 A6 B0 02F1 562 20$: MOVW 2(R6),R0 ;GET MACRO LINE LENGTH
50 04 C0 02F5 563 ADDL2 #4,R0 ;COUNT OVERHEAD BYTES
68 66 50 28 02F8 564 MOVCL3 R0,(R6),(R8) ;MOVE MACRO LINE
58 53 D0 02FC 565 MOVL R3,R8 ;UPDATE POINTER
56 51 D0 02FF 566 MOVL R1,R6 ;POINT PAST MACRO TEXT
57 56 D1 0302 567 30$: CMPL R6,R7 ;END OF EXPRESSION?
53 D7 1F 0305 568 BLSSU 10$ ;IF LSS NO
50 53 58 D0 0307 569 MOVL R8,R3 ;COPY END OF EXPR. POINTER
50 59 57 C3 030A 570 SUBL3 R7,R9,R0 ;COMPUTE LENGTH OF CODE TO MOVE
68 66 50 28 0310 571 BLEQ 40$ ;IF LEQ NOTHING
59 53 D0 0314 572 MOVCL3 R0,(R6),(R8) ;MOVE CODE
01F8 8F BA 0317 573 40$: MOVL R3,R9 ;UPDATE FRAME POINTER
05 031B 574 50$: POPR #^M<R3,R4,R5,R6,R7,R8> ;RESTORE REGISTERS
575 RSB
```

```
031C 577 .SBTTL MAC$SKP_OPR SKIP TO NEXT OPERAND OR EOL
031C 578
031C 579 :++
031C 580 : FUNCTIONAL DESCRIPTION:
031C 581 :
031C 582 : MAC$SKP_OPR WILL SCAN TO A COMMA OR END-OF-LINE. THIS IS
031C 583 : DONE WHEN AN ERROR IS DETECTED, SO AS TO PREVENT MULTIPLE
031C 584 : ERROR MESSAGES PER OPERAND. IF AN END-OF-LINE IS ENCOUNTERED
031C 585 : THE OPERAND FLAG IS CLEARED AND THE BEGINNING OF LINE FLAG IS
031C 586 : SET.
031C 587 :
031C 588 :--
031C 589
031C 590 MAC$SKP_OPR::
17 6B 0D E1 031C 591 BBC #FLG$V_OPRND,(R11),40$ ;BRANCH IF NOT IN OPERAND FIELD
0D 5A 91 0320 592 10$: CMPB R10,#CR ;YES--IS CHARACTER CR?
0A 13 0323 593 BEQL 20$ ;IF EQL YES
2C 5A 91 0325 594 CMPB R10,#^A/,/ ;NO--IS IT A COMMA?
0D 13 0328 595 BEQL 40$ ;IF EQL YES
FCD3' 30 032A 596 BSBW MAC$GETCHR ;NO--GET NEXT CHARACTER
F1 11 032D 597 BRB 10$ ;LOOK FOR NEW LINE OR COMMA
00 6B 0D E5 032F 598 20$: BBCC #FLG$V_OPRND,(R11),30$ ;CLEAR OPERAND FLAG FOR NEW LINE
00 6B 01 E2 0333 599 30$: BBSS #FLG$V_BOL,(R11),40$ ;FLAG AT BEGINNING OF LINE
05 0337 600 40$: RSB
0338 601
0338 602 MAC$SKIPSP::
20 5A 91 0338 603 10$: CMPB R10,#^A/ / ;IS IT A SPACE?
10 1A 033B 604 BGTRU 30$ ;IF GTR NO--NOT SPACE-LIKE EITHER
09 13 033D 605 BEQL 20$ ;IF EQL YES--READ NEXT CHARACTER
00000000'EF4A D5 033F 606 TSTL MAC$AL_CHRTAB[R10] ;CHECK ENTRY IN TABLE
05 12 0346 607 BNEQ 30$ ;IF NEQ NOT SPACE-LIKE CHARACTER
FCB5' 30 0348 608 20$: BSBW MAC$GETCHR ;GET NEXT CHARACTER
EB 11 034B 609 BRB 10$ ;CHECK IT OUT
05 034D 610 30$: RSB ;RETURN
034E 611
```



```
034E 613 .SBTTL MAC$CLOSE_FILES CLOSE ALL FILES
034E 614
034E 615 :++
034E 616 : FUNCTIONAL DESCRIPTION:
034E 617 :
034E 618 : THIS ROUTINE CLOSSES ALL POSSIBLY OPEN FILES
034E 619 :
034E 620 :--
034E 621
034E 622 MAC$CLOSE_FILES::
034E 623 $DISCONNECT RAB=W^MAC$INPUT_RAB ;DISCONNECT INPUT RECORD STREAM
50 0000'CF D0 0359 624 MOVL W^MAC$GL_CURINFDB,R0 ;GET PTR TO CURRENT FDB
16 6B 15 E5 035E 625 $CLOSE FAB=8(R0) ;CLOSE INPUT FILE
0368 626 BBCC #FLG$V OBJXST,(R11),10$ ;BRANCH IF NO OBJECT FILE
036C 627 $DISCONNECT RAB=W^MAC$OBJECT_RAB ;DISCONNECT OBJECT FILE STREAM
0377 628 $CLOSE FAB=W^MAC$OBJECT_FAB ;CLOSE OBJECT FILE
03 10 0382 629 10$: BSBB MAC$CLOSE_LIST ;CLOSE LISTING FILE
0045 31 0384 630 BRW MAC$CLOSE_LIB ;CLOSE LIBRARIES
0387 631 MAC$CLOSE_LIST::
16 6B 09 E5 0387 632 BBCC #FLG$V LSTXST,(R11),20$ ;BRANCH IF NO LISTING FILE
038B 633 $DISCONNECT RAB=W^MAC$LIST_RAB ;DISCONNECT LISTING FILE STREAM
0396 634 $CLOSE FAB=W^MAC$LIST_FAB ;CLOSE LISTING FILE
05 03A1 635 20$: RSB ;RETURN
03A2 636
03A2 637 :++
03A2 638 : FUNCTIONAL DESCRIPTION:
03A2 639 :
03A2 640 : THIS ROUTINE CLOSSES ALL OPEN FILES, AND DELETES THE OBJECT
03A2 641 : AND LISTING FILE. THIS SHOULD BE CALLED ONLY ON ABORTS.
03A2 642 :
03A2 643 :--
03A2 644
03A2 645 MAC$CLS_DEL_OBJ::
25 6B 15 E5 03A2 646 BBCC #FLG$V OBJXST,(R11),10$ ;BRANCH IF NO OBJECT FILE
50 0000'CF 9E 03A6 647 MOVAB W^MAC$OBJECT_RAB,R0 ;GET POINTER TO OBJECT RAB
51 3C A0 D0 03AB 648 MOVL RAB$L_FAB(R0),R1 ;GET POINTER TO FAB
00 04 A1 0F E3 03AF 649 BBCC #FAB$V_DLT,FAB$L_FOP(R1),+1 ;SET DELETE BIT IN FAB
51 DD 03B4 650 PUSHL R1 ;SAVE FAB ADDRESS
03B6 651 $DISCONNECT RAB=(R0) ;DISCONNECT RECORD ACCESS
50 8ED0 03BF 652 POPL R0 ;GET FAB ADDRESS BACK
03C2 653 $CLOSE FAB=(R0) ;CLOSE AND DELETE THE FILE
05 03CB 654 10$: RSB
```

```
03CC 656 .SBTTL MAC$CLOSE_LIB CLOSE MACRO LIBRARIES
03CC 657
03CC 658 ;++
03CC 659 ; FUNCTIONAL DESCRIPTION:
03CC 660 ;
03CC 661 ; THIS ROUTINE CLOSSES ALL MACRO LIBRARY FILES. THE FILES SHOULD
03CC 662 ; BE DISCONNECTED FROM RECORD ACCESS.
03CC 663 ;
03CC 664 ;--
03CC 665
03CC 666 MAC$CLOSE_LIB::
52 0000'CF D0 03CC 667 MOVL W^MAC$GL_MLB_QUE,R2 ;POINT TO THE FIRST MLB FDB
16 13 03D1 668 BEQL 20$ ;IF EQL NO LIBRARIES TO CLOSE
14 A2 9F 03D3 669 10$:
00000000'GF 01 FB 03D3 670 PUSHAB MLF$L CTINDEX(R2) ; Address of control table index
52 62 D0 03D6 671 CALLS #1,G^C[BR$CLOSE ; Close library file
00000000'8F 52 D1 03DD 672 MOVL MLF$L QLINK(R2),R2 ; Link to possible next library
EA 12 03E0 673 CMPL R2,#MAC$GL_MLB_QUE ;ALL DONE?
05 03E7 674 BNEQ 10$ ;IF NEQ NO
03E9 675 20$: RSB ;DONE
```



```
03EA 677 .SBTTL ALLOCATE/DEALLOCATE VIRTUAL MEMORY
03EA 678
03EA 679 :++
03EA 680 : FUNCTIONAL DESCRIPTION:
03EA 681 :
03EA 682 : THIS ROUTINE IS CALLED TO ALLOCATE 1 PAGE OF VIRTUAL MEMORY.
03EA 683 : AN ATTEMPT IS MADE TO GET MEMORY FROM THE RETURNED PAGES LIST.
03EA 684 : IF THAT FAILS, LIB$GET_VM IS CALLED TO ALLOCATE A NEW PAGE.
03EA 685 :
03EA 686 :--
03EA 687
03EA 688 MAC$ALL_1_PAGE::
50 0000'DF 0F 03EA 689 REMQUE @W^MAC$GL_FREE_LST,R0 ;TRY TO GET A PAGE FROM THE
03EF 690 ;RETURNED PAGES LIST
03EF 691 BVC 10$ ;IF V-CLEAR WE GOT ONE
00000000'GF 00000000'EF 1C 03F1 692 CALLG L^MAC$G_1_PAGE,G^LIB$GET_VM ;NONE THERE--ALLOCATE A NEW PAGE
20 50 E9 03FC 693 BLBC R0,NO_MEM ;BRANCH IF ALLOCATION FAILURE
50 0000'CF D0 03FF 694 MOVL W^MAC$GL_BASEADDR,R0 ;PICK UP THE BLOCK ADDRESS
05 0404 695 10$: RSB ;RETURN WITH BLOCK ADDRESS IN R0
0405 696
0405 697 :++
0405 698 : FUNCTIONAL DESCRIPTION:
0405 699 :
0405 700 : THIS ROUTINE IS CALLED TO DEALLOCATE 1 PAGE OF VIRTUAL MEMORY.
0405 701 : THE DEALLOCATED PAGES ARE PLACED ON A LINKED LIST POINTED TO
0405 702 : BY MAC$GL_FREE_LST.
0405 703 :
0405 704 :--
0405 705
0405 706 MAC$DEA_1_PAGE::
0000'CF 60 0E 0405 707 INSQUE (R0),W^MAC$GL_FREE_LST ;INSERT THE PAGE INTO THE FREE LIST
05 040A 708 RSB
040B 709
040B 710 :++
040B 711 : FUNCTIONAL DESCRIPTION:
040B 712 :
040B 713 : THIS ROUTINE ALLOCATES TWO CONTIGUOUS PAGES OF MEMORY.
040B 714 : THE ADDRESS IS RETURNED IN R0.
040B 715 :
040B 716 :--
040B 717
040B 718 MAC$ALL_2_PAGES::
0G0000000'GF 00000000'EF FA 040B 719 CALLG L^MAC$G_2_PAGES,G^LIB$GET_VM ;TRY TO GET THE PAGES
06 50 E9 0416 720 BLBC R0,NO_MEM ;BRANCH IF ALLOCATION ERROR
50 0000'CF D0 0419 721 MOVL W^MAC$GL_BASEADDR,R0 ;GOT IT--GET THE ADDRESS
05 041E 722 RSB
FBDE' 31 041F 723 NO_MEM: BRW MAC$ERR_NOMEM ;REPORT NO MEMORY ERROR
0422 724 ;(NO RETURN)
0422 725
0422 726 :++
0422 727 : Functional description:
0422 728 :
0422 729 : This routine deallocates a block of virtual memory. If the block
0422 730 : is 1 page it is returned to the free list; if it is >1 page the
0422 731 : block is returned to the system.
0422 732 : This routine is used to deallocate MXB blocks.
0422 733 :
```

```
0422 734 : Inputs:
0422 735 :      R0 = Address of block to deallocate
0422 736 :      Offset MXB$$_PAGES contains the size of the block in pages
0422 737 :
0422 738 :--
0422 739
0422 740 MAC$DEAL_BLOCK::
0422 741      CMPL      MXB$$_PAGES(R0),#1      : Is this block 1 page?
0422 742      BEQL      MAC$DEA_1_PAGE          : Yes if EQL
0422 743      PUSHL     R0                      : Put address on stack
0422 744      MOVL     SP,R1                  : and save stack address
0422 745      ASHL     #9,MXB$$_PAGES(R0),-(SP) : Put block size (in bytes) on stack
0422 746      MOVL     SP,R0                  : and save stack address
0422 747      PUSHL     R1                      : Form argument block
0422 748      PUSHL     R0                      : on stack
0422 749      CALLS     #2,G^LIB$FREE_VM        : Return virtual memory
0422 750      ADDL      #<2*4>,SP              : Clean up stack
0422 751      RSB
0422 752
0422 753
0422 754 :++
0422 755 : Functional description:
0422 756 :
0422 757 :      This routine allocates a block of virtual memory. The block
0422 758 :      size is rounded up to 1 page.
0422 759 :
0422 760 : Inputs:
0422 761 :      R1 = Number of bytes required
0422 762 :
0422 763 : Outputs:
0422 764 :      R0 = Address of memory block
0422 765 :      R1 = Number of pages allocated
0422 766 :
0422 767 :--
0422 768
0422 769 MAC$ALL_BLOCK::
0422 770      ADDL2     #511,R1                  : Round number of bytes to multiple
0422 771      BICL2     #511,R1                  : of 512 bytes.
0422 772      ASHL     #-9,R1,-(SP)              : Also convert to number of pages
0422 773      CMPL     (SP),#1                  : Is 1 page required?
0422 774      BNEQ     10$,                     : No if NEQ
0422 775      BSBB     MAC$ALL_1_PAGE           : Get 1 page
0422 776      BRB      20$,
0422 777 10$:
0422 778      PUSHL     R1                      : Stack bytes required
0422 779      MOVL     SP,R0                    : and save its address
0422 780      PUSHAL    W^MAC$GL_BASEADDR       : Push address to return block address
0422 781      PUSHL     R0                      : and address of bytes required
0422 782      CALLS     #2,G^LIB$GET_VM         : Get memory
0422 783      BLBC      R0,NO MEM               : Branch if error
0422 784      MOVL     W^MAC$GL_BASEADDR,R0     : Get base address of allocated block
0422 785      ADDL      #<1*4>,SP              : Clean stack
0422 786 20$:
0422 787      POPL      R1                      : Get pages allocated
0422 788      RSB
0422 789 :
0422 790 :
```

01 04 A0 D1 0422 741
DD 13 0422 742
50 DD 0422 743
51 DO 0422 744
7E 04 A0 09 78 0422 745
50 5E DO 0422 746
51 DD 0422 747
50 DD 0422 748
00000000'GF 02 FB 0422 749
5E 08 CO 0422 750
05 OS 0422 751

51 000001FF 8F CO 0422 770
51 000001FF 8F CA 0422 771
7E 51 F7 8F 78 0422 772
01 6E D1 0422 773
04 12 0422 774
8C 10 0422 775
1D 11 0422 776
51 DD 0422 777
50 5E DO 0422 778
0000'CF DF 0422 779
50 DD 0422 780
00000000'GF 02 FB 0422 781
AA 50 E9 0422 782
50 0000'CF DO 0422 783
5E 04 CO 0422 784
51 8ED0 0422 785
05 OS 0422 786
0481 787
0481 788
0481 789
0481 790

MAC\$MACSUB
V04-000

SUBROUTINES FOR VAX-11/780 ASSEMBLER^{K 12}
ALLOCATE/DEALLOCATE VIRTUAL MEMORY

16-SEP-1984 02:09:11 VAX/VMS Macro V04-00
5-SEP-1984 01:49:14 [MACRO.SRC]MACSUB.MAR;1

Page 22
(18)

0481 791 .END

MAC\$MACSUB
Symbol table

L 12
SUBROUTINES FOR VAX-11/780 ASSEMBLER

16-SEP-1984 02:09:11 VAX/VMS Macro V04-00
5-SEP-1984 01:49:14 [MACRO.SRC]MACSUB.MAR;1

Page 23
(18)

\$\$TMP1	=	00000001		
\$\$TMP2	=	00000060		
ARG\$K_SIZE	=	000003E8		
AUD\$K_SIZE	=	00000010		
BLNK	=	00000020		
CHRS\$M_COMMA CR	=	00000020		
CHRS\$M_ILL_CHR	=	00000040		
CHRS\$M_NUM_BER	=	00000010		
CHRS\$M_SPA_MSK	=	00000001		
CHRS\$M_SYM_CH1	=	00000008		
CHRS\$M_SYM_CHR	=	00000004		
CHRS\$M_SYM_DLM	=	00000002		
CHRS\$V_COMMA CR	=	00000005		
CHRS\$V_CVTLWC	=	00000061		
CHRS\$V_ILL_CHR	=	00000006		
CHRS\$V_NOCVT	=	0000007F		
CHRS\$V_NUM_BER	=	00000004		
CHRS\$V_SPA_MSK	=	00000000		
CHRS\$V_SYM_CH1	=	00000003		
CHRS\$V_SYM_CHR	=	00000002		
CHRS\$V_SYM_DLM	=	00000001		
CR	=	0000000D		
DSC\$A_POINTER	*****		X	04
DSC\$B_CLASS	*****		X	04
DSC\$B_DTYPE	*****		X	04
DSC\$K_CLASS_S	*****		X	04
DSC\$K_DTYPE-T	*****		X	04
DSC\$W_LENGTH	*****		X	04
FABS\$B_FNS	=	00000034		
FABS\$B_FSZ	=	0000003F		
FABS\$B_RFM	=	0000001F		
FABS\$C_BLN	=	00000050		
FABS\$C_VFC	=	00000003		
FABS\$L_FNA	=	0000002C		
FABS\$L_FOP	=	00000004		
FABS\$V_DLT	=	0000000F		
FF	=	0000000C		
FLG\$M_ALLCHR	=	00000001		
FLG\$M_BOL	=	00000002		
FLG\$M_CHKLPND	=	00100000		
FLG\$M_COMPEXPR	=	00000004		
FLG\$M_CONT	=	00000008		
FLG\$M_CRF	=	40000000		
FLG\$M_CRSEEN	=	00000001		
FLG\$M_DATRPT	=	00000010		
FLG\$M_DBGOUT	=	00004000		
FLG\$M_DLMSTR	=	00008000		
FLG\$M_ENDMCH	=	00000020		
FLG\$M_EVALEXPR	=	00000040		
FLG\$M_EXPOPT	=	00000080		
FLG\$M_EXTERR	=	00010000		
FLG\$M_EXTWRN	=	00020000		
FLG\$M_FIRSTLN	=	00000200		
FLG\$M_IFSTAT	=	00800000		
FLG\$M_IIF	=	00400000		
FLG\$M_INSERT	=	00000100		
FLG\$M_IRPC	=	20000000		

FLG\$M_LEXOP	=	00000002
FLG\$M_LSTXST	=	00000200
FLG\$M_MAC2COL	=	00000800
FLG\$M_MACL	=	00000800
FLG\$M_MACLTB	=	08000000
FLG\$M_MACTXT	=	00010000
FLG\$M_MEBLST	=	00001000
FLG\$M_MOREARG	=	00002000
FLG\$M_MOREINP	=	00000008
FLG\$M_NEWPND	=	00000400
FLG\$M_NOREF	=	01000000
FLG\$M_NTTYPEPC	=	00000020
FLG\$M_NULCHR	=	00040000
FLG\$M_OBJXST	=	00200000
FLG\$M_OPNDCHK	=	00000100
FLG\$M_OPRND	=	00002000
FLG\$M_OPTVFLIDX	=	00001000
FLG\$M_ORDLST	=	00020000
FLG\$M_P2	=	00004000
FLG\$M_RPTIRP	=	10000000
FLG\$M_SEQFIL	=	02000000
FLG\$M_SKAN	=	00008000
FLG\$M_SPECOP	=	00000004
FLG\$M_SPLALL	=	04000000
FLG\$M_STOIMF	=	00040000
FLG\$M_SYM2COL	=	00000400
FLG\$M_TOCFLG	=	00080000
FLG\$M_UPAFLG	=	00000010
FLG\$M_UPDFIL	=	00000080
FLG\$M_UPMARG	=	00000040
FLG\$M_XCRF	=	80000000
FLG\$V_ALLCHR	=	00000000
FLG\$V_BOL	=	00000001
FLG\$V_CHKLPND	=	00000014
FLG\$V_COMPEXPR	=	00000002
FLG\$V_CONT	=	00000003
FLG\$V_CRF	=	0000001E
FLG\$V_CRSEEN	=	00000020
FLG\$V_DATRPT	=	00000004
FLG\$V_DBGOUT	=	0000002E
FLG\$V_DLMSTR	=	0000002F
FLG\$V_ENDMCH	=	00000005
FLG\$V_EVALEXPR	=	00000006
FLG\$V_EXPOPT	=	00000007
FLG\$V_EXTERR	=	00000030
FLG\$V_EXTWRN	=	00000031
FLG\$V_FIRSTLN	=	00000029
FLG\$V_IFSTAT	=	00000017
FLG\$V_IIF	=	00000016
FLG\$V_INSERT	=	00000008
FLG\$V_IRPC	=	0000001D
FLG\$V_LEXOP	=	00000021
FLG\$V_LSTXST	=	00000009
FLG\$V_MAC2COL	=	0000002B
FLG\$V_MACL	=	0000000B
FLG\$V_MACLTB	=	0000001B
FLG\$V_MACTXT	=	00000010

MAC
V04

MAC\$MACSUB
Symbol table

M 12
SUBROUTINES FOR VAX-11/780 ASSEMBLER

16-SEP-1984 02:09:11 VAX/VMS Macro V04-00
5-SEP-1984 01:49:14 [MACRO.SRC]MACSUB.MAR;1

Page 24
(18)

```

FLGSV_MEBLST      = 0000000C
FLGSV_MOREARG     = 0000002D
FLGSV_MOREINP     = 00000023
FLGSV_NEWPND      = 0000000A
FLGSV_NOREF       = 00000018
FLGSV_NTTYPEPC    = 00000025
FLGSV_NULCHR      = 00000032
FLGSV_OBJXST      = 00000015
FLGSV_OPNDCHK     = 00000028
FLGSV_OPRND       = 0000000D
FLGSV_OPTVFLIDX   = 0000002C
FLGSV_ORDLST      = 00000011
FLGSV_P2          = 0000000E
FLGSV_RPTIRP      = 0000001C
FLGSV_SEQFIL      = 00000019
FLGSV_SKAN        = 0000000F
FLGSV_SPECOP      = 00000022
FLGSV_SPLALL      = 0000001A
FLGSV_STOIMF      = 00000012
FLGSV_SYM2COL     = 0000002A
FLGSV_TOCLFG      = 00000013
FLGSV_UPAF LG     = 00000024
FLGSV_UPDFIL      = 00000027
FLGSV_UPMARG      = 00000026
FLGSV_XCRF        = 0000001F
HASHSZ           = 0000007F
HYPHEN           = 0000002D
INPSK_BUFSIZ     = 000003E8
INTSK_BUFSIZ     = 000013F4
INTSK_BUFWRN     = 00001390
LBR$CLOSE        ***** X 04
LIB$FREE_VM      ***** X 04
LIB$GET_VM       ***** X 04
LIB$TRIM_FILESPEC ***** X 04
LSTSK_BUFSIZ     = 00000086
LSTSK_L_P_PAGE   = 0000003C
LSTSK_TITLE_SIZ  = 00000028
MAB$B_ARGNO      00000005
MAB$B_NAME       00000004
MAB$B_BLKSI Z    0000000C
MAB$B_DV PTR     00000008
MAB$B_LINK       00000000
MAB$B_DVLEN      00000006
MAC$AB_LINEBF    ***** X 04
MAC$AB_SBT_FILE  ***** X 04
MAC$AB_TMP SYM   ***** X 04
MAC$AB_TMP SPEC  00000000 R 03
MAC$ALL_1_PAGE   000003EA RG 04
MAC$ALL_2_PAGES  0000040B RG 04
MAC$ALL_BLOCK    00000444 RG 04
MAC$AL_CHRTAB    ***** X 04
MAC$AL_FTIM_DSC  ***** X 04
MAC$CK_BYT_TRU1  0000007D RG 04
MAC$CK_BYT_TRUN  0000008D RG 04
MAC$CK_SBY_TRU1  000000E7 RG 04
MAC$CK_SBY_TRUN  00000101 RG 04
MAC$CK_SWD_TRU1  00000119 RG 04

```

```

MAC$CK_SWD_TRUN  0000012F RG 04
MAC$CK_WRD_TRU1  000000B1 RG 04
MAC$CK_WRD_TRUN  000000BE RG 04
MAC$CLOSE_FILES  0000034E RG 04
MAC$CLOSE_LIB    000003CC RG 04
MAC$CLOSE_LIST   00000387 RG 04
MAC$CLS_DEL_OBJ  000003A2 RG 04
MAC$CVT_LOWER    0000019C RG 04
MAC$DEAL_BLOCK   00000422 RG 04
MAC$DEA T PAGE   00000405 RG 04
MAC$ERRORPT      ***** X 04
MAC$ERR_CONNECT  ***** X 04
MAC$ERR_NOMEM    ***** X 04
MAC$ERR_OPN_INP  ***** X 04
MAC$FAOUT        0000000D RG 04
MAC$FAOUTS       00000009 RG 04
MAC$GETCHR       ***** X 04
MAC$GL_BASEADDR  ***** X 04
MAC$GL_CURINFDB  ***** X 04
MAC$GL_EXPEND    ***** X 04
MAC$GL_EXPPTR    ***** X 04
MAC$GL_FREE_LST  ***** X 04
MAC$GL_HSHVAL    ***** X 04
MAC$GL_INTCNT    ***** X 04
MAC$GL_LINELN    ***** X 04
MAC$GL_LINEPT    ***** X 04
MAC$GL_LSB       ***** X 04
MAC$GL_MLB_QUE   ***** X 04
MAC$GL_PC        ***** X 04
MAC$GL_PSECTPTR  ***** X 04
MAC$GL_RECHDBUF  ***** X 04
MAC$GT_SCB       ***** X 04
MAC$G_T_PAGE     ***** X 04
MAC$G_2_PAGES    ***** X 04
MAC$G_LSTBUFD E S ***** X 04
MAC$HASH_SYM     0000005A RG 04
MAC$INPUT_RAB    ***** X 04
MAC$INPUT_XAB    ***** X 04
MAC$K_LIST_SIZE  ***** X 04
MAC$LAST_CHANCE  ***** X 04
MAC$LIST_FAB     ***** X 04
MAC$LIST_RAB     ***** X 04
MAC$OBJECT_FAB   ***** X 04
MAC$OBJECT_RAB   ***** X 04
MAC$OPEN_INPUT   000001AC RG 04
MAC$OPTIMIZE XPR  000002C8 RG 04
MAC$PASS_2_ERR   ***** X 04
MAC$RESCAN CH    000002B9 RG 04
MAC$SET_NEW_LSB  00000042 RG 04
MAC$SET_PC       0000002E RG 04
MAC$SKIPSP       00000338 RG 04
MAC$SKP_OPR      0000031C RG 04
MAC$SRC_KEYS     00000141 RG 04
MAC$TRUNC_ERR    000000CF R 04
MAC$WRTLST       ***** X 04
MAC$WRT_FAOUTS   00000000 RG 04
MAC$DATATRUNC    = 007D8800

```

MAC
V04

MAC\$MACSUB
Symbol table

N 12
SUBROUTINES FOR VAX-11/780 ASSEMBLER

16-SEP-1984 02:09:11 VAX/VMS Macro V04-00
5-SEP-1984 01:49:14 [MACRO.SRC]MACSUB.MAR;1

Page 25
(18)

MAC SUBSYS = 0000007D
MLFSK_BLKSI2 = 00000177
MLFSK_RSFLN = 000000FF
MLFSL_CTINDEX = 00000014
MLFSL_MCDEF = 00000008
MLFSL_QLINK = 00000000
MLFSQ_FNAMDS = 0000000C
MLFST_FNAM = 00000078
MLFSX_NAMBLK = 00000018
MNBSB_ARGCT = 00000017
MNBSB_NAME = 00000004
MNBSK_BLKSI2 = 0000001C
MNBSL_ARGP = 00000018
MNBSL_CRSYM = 00000013
MNBSL_LINK = 00000000
MNBSL_PAGC = 0000000F
MNBSL_PAGP = 0000000B
MNBSL_TXTP = 00000005
MNBSW_FLAG = 00000009
MXBSK_BLKSI2 = 00000008
MXBSL_LINK = 00000000
MXBSL_PAGES = 00000004
NAMSC_BLN = 00000060
NAMSC_MAXRSS = 000000FF
NO MEM = 0000041F R 04
OBJSK_BUFSIZ = 00000200
OPFSM_LASTOPR = 00002000
OPFSM_OPTEXP = 00001000
OPFSV_LASTOPR = 0000000D
OPFSV_OPTEXP = 0000000C
PSCSB_NAME = 00000004
PSCSB_SEG = 0000000C
PSCSB_UNUSED = 0000000B
PSCSK_BLKSI2 = 00000013
PSCSK_NO_OPTS = 0000000A
PSCSL_CURLOC = 0000000F
PSCSL_LINK = 00000000
PSCSL_MAXLGTH = 00000005
PSCSM_ABS = 00000007
PSCSM_ALIGNFLG = 00004000
PSCSM_ALLOPTNS = 000003FF
PSCSM_BYTE = 00004000
PSCSM_CON = 0000000B
PSCSM_DEFAULT = 000001C8
PSCSM_EXE = 000000C0
PSCSM_GBL = 00000010
PSCSM_LCL = 0000000F
PSCSM_LIB = 00000002
PSCSM_LONG = 00004800
PSCSM_NOEXE = 0000000B
PSCSM_NOPIC = 0000000E
PSCSM_NORD = 0000007F
PSCSM_NOSHR = 000000DF
PSCSM_NOVEC = 000000FF
PSCSM_NOWRT = 0000000F
PSCSM_OVR = 00000004
PSCSM_PAGE = 00006400

PSCSM_PIC = 00000001
PSCSM_QUAD = 00004C00
PSCSM_RD = 00000080
PSCSM_REL = 00000008
PSCSM_SHR = 00000020
PSCSM_USR = 000000FD
PSCSM_VEC = 00000200
PSCSM_WORD = 00004400
PSCSM_WRT = 00000180
PSCSV_ALIGNMENT = 00000004
PSCSV_ALIGNFLG = 0000000E
PSCSV_ALIGNMENT = 0000000A
PSCSV_EXE = 00000006
PSCSV_GBL = 00000004
PSCSV_LIB = 00000001
PSCSV_OVR = 00000002
PSCSV_PIC = 00000000
PSCSV_RD = 00000007
PSCSV_REL = 00000003
PSCSV_SHR = 00000005
PSCSV_VEC = 00000009
PSCSV_WRT = 00000008
PSCSW_FLAG = 00000009
PSCSW_OPTIONS = 0000000D
RABSL_FAB = 0000003C
RABSL_RHB = 0000002C
RDXSV_BINARY = 00000000
RDXSV_DECIMAL = 00000002
RDXSV_DOUBLE = 00000005
RDXSV_FLOAT = 00000004
RDXSV_GFLOAT = 00000006
RDXSV_HEX = 00000003
RDXSV_HFLOAT = 00000007
RDXSV_OCTAL = 00000001
REGS_PC = 0000000F
SEMI = 0000003B
STBSK_PG_MISS = 0000000A
SUM\$INIT_EDIT = ***** X 04
SYMSB_NAME = 00000004
SYMSB_SEG = 0000000C
SYMSB_TOKEN = 0000000B
SYMSK_BLKSI2 = 0000000D
SYMSK_MAXLEN = 0000001F
SYMSK_TWOCOL = 00000010
SYMSL_LINK = 00000000
SYMSL_VAL = 00000005
SYMSM_ABS = 00000010
SYMSM_ASN = 00000100
SYMSM_CRFO = 00002000
SYMSM_DEBUG = 00000020
SYMSM_DEF = 00000001
SYMSM_DELMAC = 00000200
SYMSM_EPT = 00000200
SYMSM_EXTRN = 00000008
SYMSM_GLOBL = 00000004
SYMSM_LOCAL = 00000040
SYMSM_ODBG = 00000400


```
SYMSM_REF      = 00000080
SYMSM_RELPSECT = 00000800
SYMSM_SUPR     = 00004000
SYMSM_WEAK     = 00000002
SYMSM_XCRF     = 00001000
SYMSV_ABS      = 00000004
SYMSV_ASN      = 00000008
SYMSV_CRFO     = 0000000D
SYMSV_DEBUG    = 00000005
SYMSV_DEF      = 00000000
SYMSV_DELMAC   = 00000009
SYMSV_EPT      = 00000009
SYMSV_EXTRN    = 00000003
SYMSV_GLOBL    = 00000002
SYMSV_LOCAL    = 00000006
SYMSV_ODBG     = 0000000A
SYMSV_REF      = 00000007
SYMSV_RELPSECT = 0000000B
SYMSV_SUPR     = 0000000E
SYMSV_WEAK     = 00000001
SYMSV_XCRF     = 0000000C
SYMSW_FLAG     = 00000009
SYSSASCTIM     ***** GX 04
SYSSCLOSE      ***** GX 04
SYSSCONNECT    ***** GX 04
SYSSDISCONNECT ***** GX 04
SYSSFAOL       ***** GX 04
SYSSOPEN       ***** GX 04
TAB            = 00000009
X1             = 00000400
X2             = 0000000F
XABSQ_CDT      ***** X 04
```

+-----+
! Psect synopsis !
+-----+

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
. BLANK .	00000000 (0.)	01 (1.)	NOPIC USR CON REL LCL NOSHR EXE RD WRT NOVEC BYTE
\$AB\$\$	00000177 (375.)	02 (2.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
MAC\$TEMP STOR	00000100 (256.)	03 (3.)	NOPIC USR CON REL GBL NOSHR EXE RD WRT NOVEC LONG
MAC\$RO_CODE_P15	00000481 (1153.)	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	00:00:01.32
Command processing	105	00:00:00.39	00:00:04.87
Pass 1	294	00:00:06.35	00:00:38.21
Symbol table sort	0	00:00:00.87	00:00:06.03
Pass 2	149	00:00:01.70	00:00:08.82
Symbol table output	44	00:00:00.21	00:00:00.79

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	625	00:00:09.58	00:01:00.06

The working set limit was 1500 pages.

55860 bytes (110 pages) of virtual memory were used to buffer the intermediate code.
There were 50 pages of symbol table space allocated to hold 898 non-local and 50 local symbols.
791 source lines were read in Pass 1, producing 24 object records in Pass 2.
23 pages of virtual memory were used to define 21 macros.

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

Macro library name	Macros defined
-----	-----
_\$255\$DUA28:[MACRO.OBJ]MACRO.MLB;1	8
_\$255\$DUA28:[SYSLIB]STARLET.MLB;2	14
TOTALS (all libraries)	22

990 GETS were required to define 22 macros.

There were no errors, warnings or information messages.

MACRO/LIS=LIS\$:MACSUB/OBJ=OBJ\$:MACSUB MSRC\$:MACSUB/UPDATE=(ENH\$:MACSUB)+LIB\$:MACRO/LIB

0226 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

