


```
LL      IIIIII  BBBB BBBB  MM      MM      000000  VV      VV      TTTTTTTTTT  UU      UU      CCCCCCCC
LL      IIIIII  BBBB BBBB  MM      MM      000000  VV      VV      TTTTTTTTTT  UU      UU      CCCCCCCC
LL      II      BB      BB  MMMM  MMMM  00      00  VV      VV      TT      TT      CC      CC
LL      II      BB      BB  MMMM  MMMM  00      00  VV      VV      TT      TT      CC      CC
LL      II      BB      BB  MM  MM  MM      00      00  VV      VV      TT      TT      CC      CC
LL      II      BBBB BBBB  MM      MM      00      00  VV      VV      TT      TT      CC      CC
LL      II      BBBB BBBB  MM      MM      00      00  VV      VV      TT      TT      CC      CC
LL      II      BB      BB  MM      MM      00      00  VV      VV      TT      TT      CC      CC
LL      II      BB      BB  MM      MM      00      00  VV      VV      TT      TT      CC      CC
LL      II      BB      BB  MM      MM      00      00  VV      VV      TT      TT      CC      CC
LL      II      BB      BB  MM      MM      00      00  VV      VV      TT      TT      CC      CC
LLLLLLLLLL  IIIIII  BBBB BBBB  MM      MM      000000  VV      VV      TT      TT      UU      UU      CCCCCCCC
LLLLLLLLLL  IIIIII  BBBB BBBB  MM      MM      000000  VV      VV      TT      TT      UU      UU      CCCCCCCC
```

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


(2) 82
(3) 112

DECLARATIONS
LIB\$MOVTUC - translate and move until escape found

```
0000 1 .TITLE LIB$MOVTUC - Move translated until escape character
0000 2 .IDENT /1-012/ ; File: LIBMOVTUC.MAR Edit: RKR1012
0000 3
0000 4 :
0000 5 :*****
0000 6 :*
0000 7 :* COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 8 :* DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 9 :* ALL RIGHTS RESERVED.
0000 10 :*
0000 11 :* THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 12 :* ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 13 :* INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 14 :* COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 15 :* OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 16 :* TRANSFERRED.
0000 17 :*
0000 18 :* THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 19 :* AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 20 :* CORPORATION.
0000 21 :*
0000 22 :* DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 23 :* SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 24 :*
0000 25 :*
0000 26 :*****
0000 27 :
0000 28 :
0000 29 :++
0000 30 : FACILITY: General Utility Library
0000 31 :
0000 32 : ABSTRACT:
0000 33 :
0000 34 : A translate and move of the source string to the destination
0000 35 : is performed until an escape character is found.
0000 36 :
0000 37 : ENVIRONMENT: User Mode, AST Reentrant
0000 38 :
0000 39 :--
0000 40 : AUTHOR: Donald G. Petersen, CREATION DATE: 03-Jan-78
0000 41 :
0000 42 : MODIFIED BY:
0000 43 :
0000 44 : DGP, 03-Jan-78 : VERSION 00
0000 45 : 01 - Original
0000 46 : 00-02 - DGP 06-Jan-78 - Point to escape character properly
0000 47 : 00-03 - DGP 18-Jan-78 - add optional fifth argument (ECO)
0000 48 : 1-001 - Update version number and copyright notice. JBS 16-NOV-78
0000 49 : 1-002 - Add "" to PSECT directive. JBS 21-DEC-78
0000 50 : 1-003 - Handle dynamic destination strings. JBS 20-MAR-1979
0000 51 : 1-004 - Change OTSS$ and LIB$ to STR$. JBS 21-MAY-1979
0000 52 : 1-005 - Put all externals in .EXTRN directives. JBS 19-JUN-1979
0000 53 : 1-006 - Return zero if source string exhausted with no escape.
0000 54 : SPR 11-25936 SBL 7-Sept-1979
0000 55 : 1-007 - Use general addressing when calling external routines, to
0000 56 : make this routine position independent. JBS 15-SEP-1979
0000 57 : 1-008 - Enhance to recognize additional classes of string descriptors
```



```
0000 58 : by invoking LIB$ANALYZE_SDESC_R3 to extract length and address
0000 59 : of 1st data byte from descriptor.
0000 60 : Use LIB$SGET1_DD instead of STR$GET1_DX to allocate space.
0000 61 : This should be faster and eliminates the need for establishing
0000 62 : LIB$STRT0_RET as a handler.
0000 63 : Fix error where bad results would be returned if the escape
0000 64 : character was the 65535th.
0000 65 : RKR 22-MAY-1981
0000 66 : 1-009 - Put back 1-006 fix, accidentally removed by 1-008. Analyze
0000 67 : source descriptor before allocating dest. SBL 23-Sept-1981
0000 68 : 1-010 - Add special-case code to process string descriptors that
0000 69 : "read" like fixed string descriptors.
0000 70 : Fix calling sequence to LIB$SGET1_DD. Needs length by
0000 71 : reference.
0000 72 : RKR 8-OCT-1981.
0000 73 : 1-011 - Redirect jsb's from LIB$ANALYZE_SDESC_R3 to
0000 74 : LIB$ANALYZE_SDESC_R2. Reorganize register usage to free
0000 75 : R10 which can be removed from the entry mask.
0000 76 : RKR 18-NOV-1981
0000 77 : 1-012 - Fix case where we have dynamic or varying destination,
0000 78 : escape detected, and no fill supplied. Need to adjust length
0000 79 : of destination to be actual no. of chars. translated.
0000 80 : RKR 17-AUG-1982
```

```
0000 82      .SBTTL  DECLARATIONS
0000 83      :
0000 84      : INCLUDE FILES:
0000 85      :
0000 86      $DSCDEF                      ; Descriptor symbols
0000 87      :
0000 88      : MACROS:
0000 89      :
0000 90      :     NONE
0000 91      :
0000 92      : EQUATED SYMBOLS:
0000 93      :
0000 94      :     NONE
0000 95      :
0000 96      : OWN STORAGE:
0000 97      :
0000 98      :     NONE
0000 99      :
0000 100     : PSECT DECLARATIONS:
0000 101     :
0000 102     : .PSECT _LIB$CODE PIC, SHR, LONG, EXE, NOWRT
0000 103     :
0000 104     : EXTERNALS
0000 105     :
0000 106     : .DSABL  GBL                      ; Only explicit externals
0000 107     : .EXTRN  LIB$SCOPY_R DX6          ; String copy by refrence
0000 108     : .EXTRN  LIB$SGET1-DD           ; Allocate a string
0000 109     : .EXTRN  LIB$ANALYZE_SDESC_R2    ; Extract length and address of
0000 110     :                               ; 1st data byte
```



```
0000 112 .SBTTL LIB$MOVTUC - translate and move until escape found
0000 113 :++
0000 114 : FUNCTIONAL DESCRIPTION:
0000 115 :
0000 116 : Each character of the source string is used as an index into
0000 117 : the translation table. If the translation character is the
0000 118 : escape character then the move halts.
0000 119 : Otherwise the translated character is placed into the
0000 120 : destination string.
0000 121 : Translation continues until either the source or destination
0000 122 : strings are exhausted, or an escape character is found.
0000 123 : If an escape character is found, the relative index of the
0000 124 : character in the source string which translated into the
0000 125 : escape character is returned. Otherwise; a zero is returned.
0000 126 : If either string is of zero length, then a zero is returned.
0000 127 : If the destination string cannot be allocated, or a descriptor
0000 128 : can't be handled by LIB$ANALYZE_SDESC, a -1 is returned.
0000 129 :
0000 130 : CALLING SEQUENCE:
0000 131 :
0000 132 :     esc_index.wl.v = LIB$MOVTUC (src.rt.dx,
0000 133 :                               esc.rt.dx,
0000 134 :                               table.rt.dx,
0000 135 :                               dst.wt.dx
0000 136 :                               [,fill.rt.dx])
0000 137 :
0000 138 : INPUT PARAMETERS:
0000 139 :
00000004 0000 140 :     SOURCE = 4 ; Adr of source string desc.
00000008 0000 141 :     ESC = 8 ; Adr of escape string desc
0000000C 0000 142 :     TABLE = 12 ; Adr of translation table desc
00000014 0000 143 :     FILL = 20 ; Adr of fill character desc
0000 144 :
0000 145 : IMPLICIT INPUTS:
0000 146 :
0000 147 :     NONE
0000 148 :
0000 149 : OUTPUT PARAMETERS:
0000 150 :
00000010 0000 151 :     DEST = 16 ; Adr of destination string desc
0000 152 :
0000 153 : IMPLICIT OUTPUTS:
0000 154 :
0000 155 :     NONE
0000 156 :
0000 157 : FUNCTION VALUE:
0000 158 :
0000 159 :     esc_index.wl.v
0000 160 :
0000 161 : SIDE EFFECTS:
0000 162 :
0000 163 :     May allocate storage for the destination.
0000 164 :
0000 165 : --
0000 166 :
03FC 0000 167 : .ENTRY LIB$MOVTUC , ^M<R2, R3, R4, R5, R6, R7, R8, R9>
0002 0002 168 : ; Entry point
```

```
0002 169 :+
0002 170 : Get the length and address of the source string.
0002 171 :-
50 04 AC D0 0002 172 : MOVL SOURCE(AP), R0 ; Address of SOURCE descriptor
02 03 A0 91 0006 173 : CMPB DSC$B_CLASS(R0), #DSC$K_CLASS_D ; read like fixed ?
06 1A 000A 174 : BGTRU 5$ ; no
56 04 BC 7D 000C 175 : MOVQ @SOURCE(AP), R6 ; length->R6, address->R7
0F 11 0010 176 : BRB 15$ ; join common flow
0012 177
00000000'GF 16 0012 178 5$: JSB G^LIB$ANALYZE_SDESC_R2 ; Extract: length->R1, addr->R2
03 50 E8 0018 179 : BLBS R0, 10$ ; Skip if no error
010F 31 001B 180 : BRW ERROR ; Error return
001E 181
56 51 7D 001E 182 10$: MOVQ R1, R6 ; save length and addr of SOURCE
0021 183 : length ->R6, addr ->R7
0021 184
0021 185 :+
0021 186 : If the destination string is dynamic, allocate enough space for it
0021 187 : that it will hold the whole translated source string.
0021 188 :-
50 10 AC D0 0021 189 15$: MOVL DEST(AP), R0 ; Point to dest descr.
03 A0 02 91 0025 190 : CMPB #DSC$K_CLASS_D, DSC$B_CLASS(R0) ; Dynamic?
17 12 0029 191 : BNEQ 20$ ; No, use it as is.
56 DD 002B 192 : PUSHL R6 ; length
50 DD 002D 193 : PUSHL R0 ; address
04 AE 3F 002F 194 : PUSHAW 4(SP) ; address of length
00000000'GF 02 FB 0032 195 : CALLS #2, G^LIB$SGET1_DD
5E 04 C0 0039 196 : ADDL2 #4, SP ; realign stack
03 50 E8 003C 197 : BLBS R0, 20$ ; continue if successful
00EB 31 003F 198 : BRW ERROR ; else prepare to quit
0042 199
0042 200 :+
0042 201 : Extract the various lengths and addresses we will need and leave
0042 202 : in registers for the actual MOV TUC instruction to follow.
0042 203 :-
50 0C AC D0 0042 204 20$: MOVL TABLE(AP), R0 ; Address of TABLE descriptor
02 03 A0 91 0046 205 : CMPB DSC$B_CLASS(R0), #DSC$K_CLASS_D ; read like fixed ?
06 1A 004A 206 : BGTRU 25$ ; no
58 04 A0 D0 004C 207 : MOVL DSC$A_POINTER(R0), R8 ; address of TABLE
0F 11 0050 208 : BRB 35$ ; join common flow
0052 209
00000000'GF 16 0052 210 25$: JSB G^LIB$ANALYZE_SDESC_R2 ; Extract: length->R1, addr->R2
03 50 E8 0058 211 : BLBS R0, 30$ ; Quit if error
00CF 31 005B 212 : BRW .
005E 213
58 52 D0 005E 214 30$: MOVL R2, R8 ; save addr of TABLE
0061 215
05 6C 91 0061 216 35$: CMPB (AP), #<FILL/4> ; Check for presence of fill
1F 1F 0064 217 : BLSSU 45$ ; if not there
50 14 AC D0 0066 218 : MOVL FILL(AP), R0 ; Address of FILL descriptor
02 03 A0 91 006A 219 : CMPB DSC$B_CLASS(R0), #DSC$K_CLASS_D ; read like fill ?
06 1A 006E 220 : BGTRU 40$ ; no
59 04 B0 9A 0070 221 : MOVZBL @DSC$A_POINTER(R0), R9 ; value of fill character
0F 11 0074 222 : BRB 45$ ; join common flow
0076 223
00000000'GF 16 0076 224 40$: JSB G^LIB$ANALYZE_SDESC_R2 ; Extract: length->R1, addr->R2
03 50 E8 007C 225 : BLBS R0, 42$
```



```

      00AB 31 007F 226      BRW      ERROR      ; Quit if error
59 62 9A 0082 227 42$: MOVZBL (R2), R9      ; value of fill character
      0085 228
50 08 AC D0 0085 229 45$: MOVL      ESC(AP), R0      ; Address of ESC descriptor
02 03 A0 91 0089 230      CMPB      DSC$B_CLASS(R0), #DSC$K_CLASS_D ; read like fixed ?
      008D 231      BGTRU      50$      ; no
53 04 B0 9A 008F 232      MOVZBL      @DSC$A_POINTER(R0), R3      ; value of ESC character
      0F 11 0093 233      BRB      55$      ; join common flow
      0095 234
00000000'GF 16 0095 235 50$: JSB      G^LIB$ANALYZE_SDESC_R2 ; Extract: length->R1, addr->R2
      03 50 E8 009B 236      BLBS      R0, 52$
      008C 31 009E 237      BRW      ERROR      ; Quit if error
      00A1 238
53 62 9A 00A1 239 52$: MOVZBL (R2), R3      ; value of ESC character
50 10 AC D0 00A4 240 55$: MOVL      DEST(AP), R0      ; Address of DEST descriptor
02 03 A0 91 00A8 241      CMPB      DSC$B_CLASS(R0), #DSC$K_CLASS_D ; read like fixed ?
      06 1A 00AC 242      BGTRU      60$      ; no
51 10 BC 7D 00AE 243      MOVQ      @DEST(AP), R1      ; length->R1, address->R2
      09 11 00B2 244      BRB      TEST_VS      ; join common flow
      00B4 245
00000000'GF 16 00B4 246 60$: JSB      G^LIB$ANALYZE_SDESC_R2 ; Extract: length->R1, addr->R2
      70 50 E9 00BA 247      BLBC      R0, ERROR      ; Quit if error
      00BD 248
      00BD 249 ;+
      00BD 250 ; Class_VS destination is handled as a special case. We must try to
      00BD 251 ; force its current length (CURLen) field to be the same size as the
      00BD 252 ; source length. However, this new length must be the MIN( source_len,
      00BD 253 ; and MAXSTRLEN). If source len is greater than MAXSTRLEN, then output
      00BD 254 ; will be truncated to MAXSTRLEN chars.
      00BD 255 ;-
      00BD 256 TEST_VS:
50 10 AC D0 00BD 257      MOVL      DEST(AP), R0      ; Address of DEST descriptor
0B 03 A0 91 00C1 258      CMPB      DSC$B_CLASS(R0), #DSC$K_CLASS_VS ; Class_VS ?
      11 12 00C5 259      BNEQ      75$      ; no, no special action needed
      60 56 B1 00C7 260      CMPW      R6, DSC$W_MAXSTRLEN(R0) ; SOURCE len : MAXSTRLEN
      05 15 00CA 261      BLEQ      65$      ; if SOURCE len leq
51 60 3C 00CC 262      MOVZWL      DSC$W_MAXSTRLEN(R0), R1 ; use MAXSTRLEN for CURLen
      03 11 00CF 263      BRB      70$
      00D1 264
      51 56 3C 00D1 265 65$: MOVZWL R6, R1      ; use SOURCE len for CURLen
04 B0 51 B0 00D4 266 70$: MOVW      R1, @DSC$A_POINTER(R0) ; rewrite CURLen field
      00D8 267 75$:
      00D8 268
      00D8 269 ;+
      00D8 270 ;
      00D8 271 ;
      00D8 272 ;
      00D8 273 ;
      00D8 274 ;
      00D8 275 ;
      00D8 276 ;
      00D8 277 ;
      00D8 278 ;
      00D8 279 ;
      00D8 280 ;
      00D8 281 ;
      00D8 282 ;

For those of you who have lost track,
register contents at this point:

R0 = address of DEST descriptor
R1 = length of DEST string
R2 = address of 1st byte of DEST string
R3 = value of ESC character
R4 = garbage
R5 = garbage
R6 = length of SOURCE string
R7 = address of 1st byte of SOURCE string
R8 = address of 1st byte of TABLE
R9 = value of FILL character (if supplied)
```

```

62 51 68 53 67 56 2F 00D8 283 ;+
                        00D8 284 ; prototype movtc
                        00D8 285 ; movtc srclen, srcaddr, escchar, tableaddr, dstlen, dstaddr
                        00D8 286 ; -
                        00D8 287 ; MOV TUC R6, (R7), R3, (R8), R1, (R2)
00DF 288 ; State of regs after a MOVTUC instr.
00DF 289 ; R0 = number of bytes remaining in
00DF 290 ; source string (including the
00DF 291 ; byte which caused the escape.
00DF 292 ; Is zero only if the entire source
00DF 293 ; string was translated and moved
00DF 294 ; without escape.
00DF 295 ; R1 = address of the byte which
00DF 296 ; resulted in destination string
00DF 297 ; exhaustion or escape. If no
00DF 298 ; exhaustion or escape, then
00DF 299 ; address of one byte beyond the
00DF 300 ; source string.
00DF 301 ; R2 = 0
00DF 302 ; R3 = address of the table
00DF 303 ; R4 = number of bytes remaining in the
00DF 304 ; destination string.
00DF 305 ; R5 = address of the byte in the
00DF 306 ; destination string which would
00DF 307 ; have received the translated byte
00DF 308 ; that caused the escape or would
00DF 309 ; have received a translated byte
00DF 310 ; if the source string were not
00DF 311 ; exhausted. If not exhaustion
00DF 312 ; or escape, then address of one
00DF 313 ; byte beyond the destination
00DF 314 ; string.
                        00DF 315 ; BVC 80$
                        00E1 316 ;
                        00E1 317 ;
00E1 318 ;
50 56 50 A3 00E1 319 ; SUBW3 R0, R6, R0
00E5 320 ; R0 = length of source - no. of bytes
00E5 321 ; remaining in source including escape
00E5 322 ; character
00E7 323 ; 1 - origin
00E9 324 ;
50 D4 00E9 325 80$: CLRL R0 ; We return zero if the destination
00EB 326 ; string was full before an escape found
05 6C 91 00EB 327 85$: CMPB (AP), #<FILL/4> ; Check for presence of fill character
0C 1F 00EE 328 ; branch if no fill
7E 50 D0 00F0 329 ; save index on stack
65 54 59 6E 00 2C 00F3 330 ; MOV C5 #0, 0(SP), R9, R4, (R5); fill destination string
50 8E D0 00F9 331 ; MOV L (SP)+, R0
00FC 332 ; R0 = return value
00FC 333 90$: ; No fill present, R0 all set to return
50 D5 00FC 334 ; Was escape encountered ?
01 12 00FE 335 ; yes
04 0100 336 ; no, we're all set
0101 337 ;
52 10 AC D0 0101 338 91$: MOV L DEST(AP), R2 ; addr of destination descriptor
03 A2 02 91 0105 339 ; CMPB #DSC$K_CLASS_D, DSC$B_CLASS(R2) ; was it dynamic ?
```



```
03 A2 0C 13 0109 340      BEQL 94$      ; yes
      0B 91 010B 341      CMPB #DSC$K_CLASS_VS, DSC$B_CLASS(R2) ; was it varying ?
      05 12 010F 342      BNEQ 92$      ; must have static length, exit
      0111 343
      0111 344 ; Varying destination -- must adjust length to only number of bytes
      0111 345 ; actually translated.
      0111 346 ; For varying string we can do this by rewriting the CURLEN field.
04 B2 50 01 A3 0111 347      SUBW3 #1, R0, @DSC$A_POINTER(R2) ; rewrite CURLEN field
      04 0116 348 92$: RET
      0117 349
      0117 350 ; Dynamic destination -- must adjust length to only number of bytes
      0117 351 ; actually translated.
      57 50 D0 0117 352 94$: MOVL R0, R7      ; save index to return in spare reg
      51 04 50 D7 011A 353      DECL R0      ; Number of bytes in actual result
      00000000 GF 011C 354      MCVL DSC$A_POINTER(R2), R1 ; addr of current destination
      04 50 E9 0120 355      JSB G^LIB$SCOPY_R_DX6 ; recopy to self with correct length
      50 57 D0 0126 356      BLBC R0, ERROR ; couldn't allocate temp
      04 0129 357      MOVL R7, R0      ; restore index
      012C 358      RET      ; now return
      012D 359
      012D 360 ;+
      012D 361 ; Come here if the allocation of the destination string fails or a
      012D 362 ; descriptor is bad. Return a -1, since the string position cannot
      012D 363 ; be -1.
      012D 364 ;-
      50 01 CE 012D 365 ERROR: MNEGL #1, R0      ; Return -1
      04 0130 366      RET      ; to the caller.
      0131 367
      0131 368      .END
```

LIB\$MOV TUC
Symbol table

DEST	=	00000010		
DSC\$A_POINTER	=	00000004		
DSC\$B_CLASS	=	00000003		
DSC\$K_CLASS_D	=	00000002		
DSC\$K_CLASS_VS	=	0000000B		
DSC\$W_MAXSTRLEN	=	00000000		
ERROR	=	0000012D	R	02
ESC	=	00000008		
FILL	=	00000014		
LIB\$ANALYZE_SDESC_R2	*****		X	02
LIB\$MOV TUC	00000000		RG	02
LIB\$SCOPY_R_DX6	*****		X	02
LIB\$SGET1_DD	*****		X	02
SOURCE	=	00000004		
TABLE	=	0000000C		
TEST_VS	000000BD		R	02

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

PSECT name	Allocation	PSECT No.	Attributes															
. ABS .	00000000 (0.)	00 (0.)	NOPIC	USR	CON	ABS	LCL	NOSHR	NOEXE	NORD	NOWRT	NOVEC	BYTE					
\$AB\$\$	00000000 (0.)	01 (1.)	NOPIC	USR	CON	ABS	LCL	NOSHR	EXE	RD	WRT	NOVEC	BYTE					
_LIB\$CODE	00000131 (305.)	02 (2.)	PIC	USR	CON	REL	LCL	SHR	EXE	RD	NOWRT	NOVEC	LONG					

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

Phase	Page faults	CPU Time	Elapsed Time
Initialization	33	00:00:00.05	00:00:01.32
Command processing	132	00:00:00.30	00:00:01.20
Pass 1	138	00:00:01.39	00:00:07.04
Symbol table sort	0	00:00:00.13	00:00:00.38
Pass 2	76	00:00:00.55	00:00:03.17
Symbol table output	4	00:00:00.02	00:00:00.02
Psect synopsis output	2	00:00:00.01	00:00:00.17
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	387	00:00:02.45	00:00:13.35

The working set limit was 1200 pages.
11543 bytes (23 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 142 non-local and 23 local symbols.
368 source lines were read in Pass 1, producing 13 object records in Pass 2.
8 pages of virtual memory were used to define 7 macros.

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

Macro library name	Macros defined
_\$255\$DUA28:[SYSLIB]STARLET.MLB;2	4

LIB\$MOVTUC
VAX-11 Macro Run Statistics

- Move translated until escape character L 12
16-SEP-1984 00:14:35 VAX/VMS Macro V04-00 Page 10
6-SEP-1984 11:09:23 [LIBRTL.SRC]LIBMOVTUC.MAR;1 (3)

190 GETS were required to define 4 macros.

There were no errors, warnings or information messages.

MACRO/ENABLE=SUPPRESSION/DISABLE=(GLOBAL,TRACEBACK)/LIS=LIS\$:LIBMOVTUC/OBJ=OBJ\$:LIBMOVTUC MSRC\$:LIBMOVTUC/UPDATE=(ENH\$:LIBMOVTUC)

0208 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

