D 8

```
LLL                 IIIIIIIII      BBBBBBBBBBBB     RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
LLL                 IIIIIIIII      BBBBBBBBBBBB     RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
LLL                 IIIIIIIII      BBBBBBBBBBBB     RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
LLL                    III         BBB         BBB  RRR         RRR        TTT        LLL
LLL                    III         BBB         BBB  RRR         RRR        TTT        LLL
LLL                    III         BBB         BBB  RRR         RRR        TTT        LLL
LLL                    III         BBB         BBB  RRR         RRR        TTT        LLL
LLL                    III         BBB         BBB  RRR         RRR        TTT        LLL
LLL                    III         BBB         BBB  RRR         RRR        TTT        LLL
LLL                    III         BBBBBBBBBBBB     RRRRRRRRRRRR           TTT        LLL
LLL                    III         BBBBBBBBBBBB     RRRRRRRRRRRR           TTT        LLL
LLL                    III         BBBBBBBBBBBB     RRRRRRRRRRRR           TTT        LLL
LLL                    III         BBB         BBB  RRR   RRR              TTT        LLL
LLL                    III         BBB         BBB  RRR   RRR              TTT        LLL
LLL                    III         BBB         BBB  RRR   RRR              TTT        LLL
LLL                    III         BBB         BBB  RRR      RRR           TTT        LLL
LLL                    III         BBB         BBB  RRR      RRR           TTT        LLL
LLL                    III         BBB         BBB  RRR      RRR           TTT        LLL
LLLLLLLLLLLLLLL     IIIIIIIII      BBBBBBBBBBBB     RRR         RRR        TTT        LLLLLLLLLLLLLLL
LLLLLLLLLLLLLLL     IIIIIIIII      BBBBBBBBBBaB     RRR         RRR        TTT        LLLLLLLLLLLLLLL
LLLLLLLLLLLLLLL     IIIIIIIII      BBBBBBBBBBBB     RRR         RRR        TTT        LLLLLLLLLLLLLLL
```

```
_$

Sy
--
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI
LI

LI
LI
LI
LI
LI
LI
LI

LI
LI
LI

LI
LI
LI
LI
LI
LI
```

FILEID**LIBTRAE2A

```
LL            IIIIII    BBBBBBBB    TTTTTTTTTT  RRRRRRRR      AAAAAA    EEEEEEEEEE    222222      AAAAAA
LL            IIIIII    BBBBBBBB    TTTTTTTTTT  RRRRRRRR      AAAAAA    EEEEEEEEEE    222222      AAAAAA
LL              II      BB      BB      TT      RR      RR  AA      AA  EE          22      22  AA      AA
LL              II      BB      BB      TT      RR      RR  AA      AA  EE          22      22  AA      AA
LL              II      BB      BB      TT      RR      RR  AA      AA  EE                  22  AA      AA
LL              II      BB      BB      TT      RR      RR  AA      AA  EE                  22  AA      AA
LL              II      BBBBBBBB        TT      RRRRRRRR    AA      AA  EEEEEEEE          22    AA      AA
LL              II      BBBBBBBB        TT      RRRRRRRR    AA      AA  EEEEEEEE          22    AA      AA
LL              II      BB      BB      TT      RR  RR      AAAAAAAAAA  EE              22      AAAAAAAAAA
LL              II      BB      BB      TT      RR  RR      AAAAAAAAAA  EE              22      AAAAAAAAAA
LL              II      BB      BB      TT      RR    RR    AA      AA  EE            22        AA      AA      ....
LL              II      BB      BB      TT      RR    RR    AA      AA  EE            22        AA      AA      ....
LLLLLLLLLL    IIIIII    BBBBBBBB        TT      RR      RR  AA      AA  EEEEEEEEEE  2222222222  AA      AA      ....
LLLLLLLLLL    IIIIII    BBBBBBBB        TT      RR      RR  AA      AA  EEEEEEEEEE  2222222222  AA      AA      ....

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
```

LI

```
0000        .TITLE  LIB$TRA_EBC_ASC - Translate EBCDIC string to ASCII string
0000        .IDENT  /1-008/                  ; File: LIBTRAE2A.MAP RKR1008
0000
0000  ;
0000  ;****************************************************************************
0000  ;*                                                                          *
0000  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000  ;*  ALL RIGHTS RESERVED.                                                    *
0000  ;*                                                                          *
0000  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000  ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000  ;*  TRANSFERRED.                                                            *
0000  ;*                                                                          *
0000  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000  ;*  CORPORATION.                                                            *
0000  ;*                                                                          *
0000  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000  ;*                                                                          *
0000  ;*                                                                          *
0000  ;****************************************************************************
0000  ;
0000
0000  ;++
0000  ; FACILITY: General Utility Library
0000  ;
0000  ; ABSTRACT:
0000  ;
0000  ;       Translates an EBCDIC string to an ASCII string
0000  ;
0000  ; ENVIRONMENT: User Mode, AST Reentrant
0000  ;
0000  ;--
0000  ; AUTHOR: R. Reichert,  CREATION DATE: 03-DEC-1979
0000  ;
0000  ; MODIFIED BY:
0000  ;
0000  ; REVISION HISTORY:
0000  ;
0000  ; 1-001 - Original. RKR 03-DEC-79
0000  ; 1-002 - Change EBCDIC string descriptor documentation to "bu.dx"
0000  ;         RKR 05-DEC-79
0000  ; 1-003 - Correct .SBTTL text.  RKR 19-DEC-79
0000  ; 1-004 - Use a handler to convert STR$ signals to LIB$ status codes.
0000  ;           JBS 22-JAN-1980
0000  ; 1-005 - Refetch destination length and address after call to STR$GET1_DX.
0000  ;         SPR 11-38343 SBL 5-June-1981
0000  ; 1-006 - Enhance to recognize additional classes of string descriptors
0000  ;         by invoking LIB$ANALYZE_SDESC_R3 to extract length and
0000  ;         address of 1st data byte.
0000  ;         Redirect call to STR$GET1_DX to LIB$SGET1_DD and remove use of
0000  ;         handler.
```

```
     0000    58 ;           RKR 26-MAY-1981.
     0000    59 ; 1-007 - Add special-case code to process string descriptors that
     0000    60 ;           "read" like fixed string descriptors.  RKR 7-OCT-1981.
     0000    61 ; 1-008 - Redirect jsb's from LIB$ANALYZE_SDESC_R3 to
     0000    62 ;           LIB$ANALYZE_SDESC_R2.  18-NOV-1981.
```

```
                    0000          .SBTTL  DECLARATIONS
                    0000   ;
                    0000   ; INCLUDE FILES: NONE
                    0000   ;
                    0000
                    0000   ;
                    0000   ; EXTERNAL DECLARATIONS:
                    0000   ;
                    0000          .DSABL  GBL                              ; Prevent undeclared
                    0000                                                   ; symbols from being
                    0000                                                   ; automatically global.
                    0000
                    0000          .EXTRN  LIB$SGET1_DD           ; dynamic string allocator
                    0000
                    0000          .EXTRN  LIB$ANALYZE_SDESC_R2   ; Extract length and address of
                    0000                                         ; 1st data byte from descriptor
                    0000
                    0000          .EXTRN  LIB$AB_EBC_ASC         ; EBCDIC to ASCII translation
                    0000                                         ; table
                    0000
                    0000          .EXTRN  LIB$_INVCHA            ; Return code "invalid
                    0000                                         ; character"
                    0000
                    0000          .EXTRN  LIB$_INVARG            ; Return code "invalid
                    0000                                         ; argument"
                    0000
                    0000   ;
                    0000   ; MACROS:
                    0000   ;
                    0000
                    0000          $SSDEF                         ; Definition of SS$_ symbols
                    0000          $DSCDEF                        ; Descriptor components
                    0000   ;
                    0000   ; EQUATED SYMBOLS:
                    0000   ;
0000005C            0000   ESC_CHAR        = ^X5C          ; ASCII Character "\"
000000E0            0000   SOURCE_CODE     = ^XE0          ; EBCDIC Character "\"
                    0000
                    0000
                    0000   ;
                    0000   ; PSECT DECLARATIONS:
                    0000   ;
                00000000          .PSECT _LIB$CODE PIC, USR, CON, REL, LCL, SHR, -
                    0000                          EXE, RD, NOWRT, LONG
                    0000
```

```
0000          .SBTTL  LIB$TRA_EBC_ASC - Translate EBCDIC to ASCII
0000    ;++
0000    ; FUNCTIONAL DESCRIPTION:
0000    ;
0000    ;       Translates an EBCDIC string to a ASCII string
0000    ;       If destination string is a fixed string, its length must match
0000    ;       the length of the input string (no filling is done).
0000    ;       If destination string is varying, its length is forced
0000    ;       (if possible) to be the length of the source string.  If it
0000    ;       cannot, (MAXSTRLEN too small), LIB$_INVARG is returned.
0000    ;
0000    ; CALLING SEQUENCE:
0000    ;
0000    ;       status.wlc.v = LIB$TRA_EBC_ASC ( SRC_STR.rbu.dx, DST_STR.wt.dx)
0000    ;
0000    ; INPUT PARAMETERS:
0000    ;
0000    ;       SRC_STR.rbu.dx  address of source string descriptor (EBCDIC)
0000    ;
0000    ;       DST_STR.wt.dx   address of destination string descriptor (ASCII)
0000    ;
0000    ;
0000    ; IMPLICIT INPUTS:
0000    ;
0000    ;       The EBCDIC to ASCII translation table at LIB$AB_EBC_ASC
0000    ;
0000    ; OUTPUT PARAMETERS:
0000    ;
0000    ;       NONE
0000    ;
0000    ; IMPLICIT OUTPUTS:
0000    ;
0000    ;       NONE
0000    ;
0000    ; COMPLETION CODES:
0000    ;
0000    ;       SS$_NORMAL      - Routine successfully completed.
0000    ;
0000    ;       LIB$_INVCHA     - One or more occurences of an untranslatable
0000    ;                         character has been detected in the course
0000    ;                         of translation.
0000    ;
0000    ;       LIB$_INVARG     - If destination string is fixed string and
0000    ;                         its length is not the same as the source
0000    ;                         length.
0000    ;
0000    ;
0000    ; SIDE EFFECTS:
0000    ;
0000    ;       NONE
0000    ;
0000    ;--
0000
```

```
                          0000     ; DISPLACEMENTS OFF AP OF INPUT ARGUMENTS:
                          0000
                 00000004 0000     SRC_STR =       4
                 00000008 0000     DST_STR =       8
                          0000
                          0000
                     40FC 0000            .ENTRY LIB$TRA_EBC_ASC, ^M<IV, R2, R3, R4, R5, R6, R7>
                          0002     ;+
                          0002     ; Extract the lengths and addresses we will need from the source and
                          0002     ; destination descriptors.
                          0002     ;-
       50    04 AC     D0 0002            MOVL    SRC_STR(AP), R0           ; Address of src descriptor
       02    03 A0     91 0006            CMPB    DSC$B_CLASS(R0), #DSC$K_CLASS_D ; read like fixed ?
                06     1A 000A            BGTRU   1$                        ; no
       56    04 BC     7D 000C            MOVQ    @SRC_STR(AP), R6          ; length->R6, address->R7
                09     11 0010            BRB     2$                        ; join common flow
                          0012
      00000000'GF      16 0012     1$:    JSB     G^LIB$ANALYZE_SDESC_R2    ; Extract: length->R1, addr->R2
          56    51     7D 0018            MOVQ    R1, R6                    ; length->R6, address->R7
                          001B
       50    08 AC     D0 001B     2$:    MOVL    DST_STR(AP), R0 ; Address of dst descriptor
       02    03 A0     91 001F            CMPB    DSC$B_CLASS(R0), #DSC$K_CLASS_D ; read like fixed ?
                06     1A 0023            BGTRU   3$                        ; no
       54    08 BC     7D 0025            MOVQ    @DST_STR(AP), R4          ; length->R4, address->R5
                09     11 0029            BRB     4$                        ; join common flow
                          002B
      00000000'GF      16 002B     3$:    JSB     G^LIB$ANALYZE_SDESC_R2    ; Extract: length->R1, addr->R2
          54    51     7D 0031            MOVQ    R1, R4                    ; length->R4, address->R5
                          0034
                          0034     ;+
                          0034     ; If destination is a dynamic string, reallocate a dynamic string which
                          0034     ; is the same length as the input string.
                          0034     ;-
                          0034
       52    08 AC     D0 0034     4$:    MOVL    DST_STR(AP), R2           ; address of dst descriptor
       02    03 A2     91 0038            CMPB    DSC$B_CLASS(R2), #DSC$K_CLASS_D ; dynamic string dest ?
                1C     12 003C            BNEQ    5$                        ; not dynamic
          7E    56     3C 003E            MOVZWL  R6, -(SP)                 ; length of source
             08 AC     DD 0041            PUSHL   DST_STR(AP)               ; caller's dest desc addr
             04 AE     DF 0044            PUSHAL  4(SP)                     ; address of length of source
 00000000'GF    02     FB 0047            CALLS   #2, G^LIB$SGET1_DD        ; allocate a dynamic string
                          004E                                              ; equal in len to source
          5E    04     C0 004E            ADDL2   #4, SP                    ; restore stack
          54    56     3C 0051            MOVZWL  R6, R4                    ; dest length = source length
       55    04 A2     D0 0054            MOVL    DSC$A_POINTER(R2), R5     ; new destination address
                21     11 0058            BRB     10$                       ; join common flow
                          005A     ;+
                          005A     ; If destination is a varying string, force its CURLEN to the MIN (
                          005A     ; MAXSTRLEN, length of source).  If we can't, return LIB$_INVARG.
                          005A     ;-
                          005A     5$:
       0B    03 A2     91 005A            CMPB    DSC$B_CLASS(R2), #DSC$K_CLASS_VS ; varying string dest ?
                0E     12 005E            BNEQ    6$                        ; not varying
          56    62     B1 0060            CMPW    DSC$W_MAXSTRLEN(R2), R6 ; MAXSTRLEN ? length of source
                0E     1F 0063            BLSSU   9$                        ; won't fit, exit with error
          54    56     3C 0065            MOVZWL  R6, R4                    ; Set dest len to source length
       04 B2    54     B0 0068            MOVW    R4, @DSC$A_POINTER(R2)    ; Set CURLEN to new length
```

```
                   0D   11  006C          BRB     10$                             ; join common flow
                            006E      ;+
                            006E      ; Otherwise, we have a destination string with fixed-length semantics.
                            006E      ; Its length must match the length of the input string,
                            006E      ; else LIB$_INVARG is returned.
                            006E      ;-
                            006E      6$:
              54   56   B1  006E          CMPW    R6, R4                          ; source len = dest len ?
                   08   13  0071          BEQL    10$                             ; ok if lengths match
     50   00000000'8F   D0  0073      9$: MOVL    #LIB$_INVARG, R0                ; return "invalid argument"
                        04  007A          RET
                            007B
                            007B      ;+
                            007B      ; Set up registers for the MOVTUC and MOVTC to follow
                            007B      ;-
                            007B      10$:
              50   56   B0  007B          MOVW    R6, R0                          ; get source length in R0
              51   57   D0  007E          MOVL    R7, R1                          ; address of caller's source
                            0081      ;+
                            0081      ; registers at this point...
                            0081      ;
                            0081      ;           R0      length of source string
                            0081      ;           R1      address of source string
                            0081      ;           R2      address of destination descriptor
                            0081      ;           R3      unknown
                            0081      ;           R4      length of destination string (must be equal to R6)
                            0081      ;           R5      address of destination string
                            0081      ;           R6      length of source string (must be equal to R4)
                            0081      ;           R7      address of source string (same as R1)
                            0081      ;-
```

```
                                        0081  ;+
                                        0081  ; Actual translation loop.
                                        0081  ; repeated until we translate all error free, or fall through with
                                        0081  ; an error and perform one final MOVTC to complete translation
                                        0081  ;-
                                        0081
                                        0081  15$:
 00000000'GF    5C 8F    61    50   2F  0081          MOVTUC  RO, (R1), #ESC_CHAR, G^LIB$AB_EBC_ASC, R4, (R5)
                         65    54       008B
                                        008D                                           ; State of regs after a MOVTUC instr.
                                        008D                                           ; R0 = number of bytes remaining in
                                        008D                                           ;      source string (including the
                                        008D                                           ;      byte which caused the escape.
                                        008D                                           ;      Is zero only if the entire source
                                        008D                                           ;      string was translated and moved
                                        008D                                           ;      without escape.
                                        008D                                           ; R1 = address of the byte which
                                        008D                                           ;      resulted in destination string
                                        008D                                           ;      exhaustion or escape.  If no
                                        008D                                           ;      exhaustion or escape, then
                                        008D                                           ;      address of one byte beyond the
                                        008D                                           ;      source string.
                                        008D                                           ; R2 = 0
                                        008D                                           ; R3 = address of the table
                                        008D                                           ; R4 = number of bytes remaining in the
                                        008D                                           ;      destinatin string.
                                        008D                                           ; R5 = address of the byte in the
                                        008D                                           ;      destination string which would
                                        008D                                           ;      have received the translated byte
                                        008D                                           ;      that caused the escape or would
                                        008D                                           ;      have received a translated byte
                                        008D                                           ;      if the source string were not
                                        008D                                           ;      exhausted.  If not exhaustion
                                        008D                                           ;      or escape, then address of one
                                        008D                                           ;      byte beyond the destination
                                        008D                                           ;      string.
                               21   1C  008D          BVC     GOOD_COMPL                ; terminated by end of string
                                        008F
               85    5C 8F     90       008F          MOVB    #ESC_CHAR, (R5)+          ; store esc_char in output
                                        0093                                            ; bumping R5 to next byte pos.
                               50   D7  0093          DECL    R0                        ; adjust input count for one
                                        0095                                            ; done by hand
                               54   D7  0095          DECL    R4                        ; adjust output count for one
                                        0097                                            ; done by hand
               E0 8F     81    91       0097          CMPB    (R1)+, #SOURCE_CODE       ; was the input an escape char
                         E4    13       009B          BEQL    15$                       ; yes -- treat as success and
                                        009D                                            ; and continue
                                        009D
                                        009D  ;+
                                        009D  ; an untranslatable input char has been detected.  Translate rest of
                                        009D  ; string and exit with an error status of LIB$_INVCHA
                                        009D  ;-
                                        009D
   00000000'GF    6E     61    50   2E  009D          MOVTC   R0, (R1), 0(SP), G^LIB$AB_EBC_ASC, R4, (R5)
                         65    54       00A6
                                        00A8                                            ; NOTE: lengths are guarenteed
                                        00A8                                            ; to be equal, so fill_char is
```

```
                           00A8                                                   ; not needed.
                           00A8                                                   ; NOTE: lengths are guaranteed
                           00A8                                                   ; to be equal, so fill_char is
                           00A8                                                   ; not needed.
                           00A8                                           ; State of regs after a MOVTC instr.
                           00A8                                           ; R0 = number of translated bytes
                           00A8                                           ;      remaining in source string.
                           00A8                                           ;      Is non zero only if source string
                           00A8                                           ;      is longer than destination
                           00A8                                           ;      string.
                           00A8                                           ; R1 = address of one byte beyond the
                           00A8                                           ;      last byte in source string that
                           00A8                                           ;      was translated.
                           00A8                                           ; R2 = 0
                           00A8                                           ; R3 = address of the translation table
                           00A8                                           ; R4 = 0
                           00A8                                           ; R5 = address of one byte beyond the
                           00A8                                           ;      destination string.
                           00A8                                           ;
      50   00000000'8F  D0 00A8               MOVL    #LIB$_INVCHA, R0        ; return "invalid char" code
                        04 00AF               RET
                           00B0
                           00B0       GOOD_COMPL:
             50    01   D0 00B0               MOVL    #SS$_NORMAL, R0         ; return success
                        04 00B3               RET
                           00B4
                           00B4               .END
```

```
DSC$A_POINTER                   = 00000004
DSC$B_CLASS                     = 00000003
DSC$K_CLASS_D                   = 00000002
DSC$K_CLASS_VS                  = 0000000B
DSC$W_MAXSTRLEN                 = 00000000
DST_STR                         = 00000008
ESC_CHAR                        = 0000005C
GOOD_COMPL                        000000B0 R       02
LIB$AB_EBC_ASC                    ********     X   00
LIB$ANALYZE_SDESC_R2              ********     X   00
LIB$SGET1_DD                      ********     X   00
LIB$TRA_EBC_ASC                   00000000 RG      02
LIB$_INVARG                       ********     X   00
LIB$_INVCHA                       ********     X   00
SOURCE_CODE                     = 000000E0
SRC_STR                         = 00000004
SS$_NORMAL                      = 00000001

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

PSECT name                       Allocation           PSECT No.  Attributes
----------                       ----------           ---------  ----------
.  ABS  .                        00000000  (    0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
$ABS$                            00000000  (    0.)  01 (  1.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD     WRT NOVEC BYTE
_LIB$CODE                        000000B4  (  180.)  02 (  2.)    PIC   USR   CON   REL   LCL   SHR   EXE   RD   NOWRT NOVEC LONG

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

Phase                     Page faults    CPU Time       Elapsed Time
-----                     -----------    --------       ------------
Initialization                     31    00:00:00.04    00:00:02.29
Command processing                124    00:00:00.32    00:00:02.33
Pass 1                            208    00:00:03.42    00:00:15.02
Symbol table sort                   0    00:00:00.55    00:00:02.41
Pass 2                             70    00:00:00.79    00:00:04.03
Symbol table output                 4    00:00:00.02    00:00:00.02
Psect synopsis output               2    00:00:00.01    00:00:00.01
Cross-reference output              0    00:00:00.00    00:00:00.00
Assembler run totals              441    00:00:05.16    00:00:26.12

The working set limit was 1200 pages.
28498 bytes (56 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 549 non-local and 9 local symbols.
333 source lines were read in Pass 1, producing 13 object records in Pass 2.
9 pages of virtual memory were used to define 8 macros.
```

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

Macro library name                          Macros defined
------------------                          --------------
_$255$DUA28:[SYSLIB]STARLET.MLB;2                     5

604 GETS were required to define 5 macros.

There were no errors, warnings or information messages.

MACRO/ENABLE=SUPPRESSION/DISABLE=(GLOBAL,TRACEBACK)/LIS=LIS$:LIBTRAE2A/OBJ=OBJ$:LIBTRAE2A MSRC$:LIBTRAE2A/UPDATE=(ENH$:LIBTRAE2A)
```

LIB

0210 AH-BT13A-SE DIGITAL EQUIPMENT CORPORATION
VAX/VMS V4.0 CONFIDENTIAL AND PROPRIETARY

LIBSPAWN
LIS

LIBSTATVM
LIS

LIBTRAA2E
LIS

LIBSPANC
LIS

LIBSYMBOL
LIS

LIBTRNLOG
LIS

LIBSKPC
LIS

LIBTIMER
LIS

LIBTPARSE
LIS

LIBTRIMFI
LIS

LIBSTRRET
LIS

LIBTRAE2A
LIS