

[illegible]

3

Sy

LI

LI
LI

LI

LI
LI

LI

LI
LILI
LILI
LI

LI

LI
LI

LI

LI
LI

LI

LI
LI

LI

11

LI

LI
LI

LI

LI
LI

21

LI

LI
LI

22

LI
LI

LI

LI
LI

LI

1

[illegible]

(2) 64
(3) 110

DECLARATIONS
LIB\$TRA_EBC_ASC - Translate EBCDIC to ASCII

```
0000 1 .TITLE LIB$TRA_EBC_ASC - Translate EBCDIC string to ASCII string
0000 2 .IDENT /1-008/ ; File: LIBTRAE2A.MAR RKR1008
0000 3
0000 4
0000 5 *****
0000 6
0000 7 *
0000 8 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
0000 9 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
0000 10 * ALL RIGHTS RESERVED.
0000 11 *
0000 12 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
0000 13 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
0000 14 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
0000 15 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
0000 16 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
0000 17 * TRANSFERRED.
0000 18 *
0000 19 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
0000 20 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
0000 21 * CORPORATION.
0000 22 *
0000 23 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
0000 24 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
0000 25 *
0000 26 *****
0000 27
0000 28
0000 29 ++
0000 30 FACILITY: General Utility Library
0000 31
0000 32 ABSTRACT:
0000 33
0000 34 Translates an EBCDIC string to an ASCII string
0000 35
0000 36 ENVIRONMENT: User Mode, AST Reentrant
0000 37
0000 38 --
0000 39 AUTHOR: R. Reichert, CREATION DATE: 03-DEC-1979
0000 40
0000 41 MODIFIED BY:
0000 42
0000 43 REVISION HISTORY:
0000 44
0000 45 1-001 - Original. RKR 03-DEC-79
0000 46 1-002 - Change EBCDIC string descriptor documentation to 'bu.dx'
0000 47 RKR 05-DEC-79
0000 48 1-003 - Correct .SBTTL text. RKR 19-DEC-79
0000 49 1-004 - Use a handler to convert STR$ signals to LIB$ status codes.
0000 50 JBS 22-JAN-1980
0000 51 1-005 - Refetch destination length and address after call to STR$GET1_DX.
0000 52 SPR 11-38343 SBL 5-June-1981
0000 53 1-006 - Enhance to recognize additional classes of string descriptors
0000 54 by invoking LIB$ANALYZE_SDESC_R3 to extract length and
0000 55 address of 1st data byte.
0000 56 Redirect call to STR$GET1_DX to LIB$GET1_DD and remove use of
0000 57 handler.
```

LIB\$TRA_EBC_ASC
1-008

D 15
- Translate EBCDIC string to ASCII strin 16-SEP-1984 00:22:05 VAX/VMS Macro V04-00 Page 2
6-SEP-1984 11:11:53 [LIBRTL.SRC]LIBTRAE2A.MAR;1 (1)

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 64 .SBTTL DECLARATIONS
0000 65 :
0000 66 : INCLUDE FILES: NONE
0000 67 :
0000 68 :
0000 69 :
0000 70 : EXTERNAL DECLARATIONS:
0000 71 :
0000 72 .DSABL GBL ; Prevent undeclared
0000 73 ; symbols from being
0000 74 ; automatically global.
0000 75
0000 76 .EXTRN LIB$SGET1_DD ; dynamic string allocator
0000 77
0000 78 .EXTRN LIB$ANALYZE_SDESC_R2 ; Extract length and address of
0000 79 ; 1st data byte from descriptor
0000 80
0000 81 .EXTRN LIB$AB_EBC_ASC ; EBCDIC to ASCII translation
0000 82 ; table
0000 83
0000 84 .EXTRN LIB$_INVCHA ; Return code "invalid
0000 85 ; character"
0000 86
0000 87 .EXTRN LIB$_INVARG ; Return code "invalid
0000 88 ; argument"
0000 89
0000 90 :
0000 91 : MACROS:
0000 92 :
0000 93
0000 94 $SSDEF ; Definition of SS$ symbols
0000 95 $DSCDEF ; Descriptor components
0000 96 :
0000 97 : EQUATED SYMBOLS:
0000 98 :
0000005C 0000 99 ESC CHAR = ^X5C ; ASCII Character "\
000000E0 0000 100 SOURCE_CODE = ^XE0 ; EBCDIC Character "\
0000 101
0000 102
0000 103 :
0000 104 : PSECT DECLARATIONS:
0000 105 :
00000000 0000 106 .PSECT _LIB$CODE PIC, USR, CON, REL, LCL, SHR, -
0000 107 EXE, RD, NOWRT, LONG
0000 108
```

```
0000 110 .SBTTL LIB$TRA_EBC_ASC - Translate EBCDIC to ASCII
0000 111 :++
0000 112 : FUNCTIONAL DESCRIPTION:
0000 113 :
0000 114 : Translates an EBCDIC string to a ASCII string
0000 115 : If destination string is a fixed string, its length must match
0000 116 : the length of the input string (no filling is done).
0000 117 : If destination string is varying, its length is forced
0000 118 : (if possible) to be the length of the source string. If it
0000 119 : cannot, (MAXSTRLEN too small), LIB$_INVARG is returned.
0000 120 :
0000 121 : CALLING SEQUENCE:
0000 122 :
0000 123 : status.wlc.v = LIB$TRA_EBC_ASC ( SRC_STR.rbu.dx, DST_STR.wt.dx)
0000 124 :
0000 125 : INPUT PARAMETERS:
0000 126 :
0000 127 : SRC_STR.rbu.dx address of source string descriptor (EBCDIC)
0000 128 :
0000 129 : DST_STR.wt.dx address of destination string descriptor (ASCII)
0000 130 :
0000 131 :
0000 132 : IMPLICIT INPUTS:
0000 133 :
0000 134 : The EBCDIC to ASCII translation table at LIB$AB_EBC_ASC
0000 135 :
0000 136 : OUTPUT PARAMETERS:
0000 137 :
0000 138 : NONE
0000 139 :
0000 140 : IMPLICIT OUTPUTS:
0000 141 :
0000 142 : NONE
0000 143 :
0000 144 : COMPLETION CODES:
0000 145 :
0000 146 : SSS$_NORMAL - Routine successfully completed.
0000 147 :
0000 148 : LIB$_INVCHA - One or more occurrences of an untranslatable
0000 149 : character has been detected in the course
0000 150 : of translation.
0000 151 :
0000 152 : LIB$_INVARG - If destination string is fixed string and
0000 153 : its length is not the same as the source
0000 154 : length.
0000 155 :
0000 156 :
0000 157 : SIDE EFFECTS:
0000 158 :
0000 159 : NONE
0000 160 :
0000 161 :--
0000 162 :
```

```
0000 164 ; DISPLACEMENTS OFF AP OF INPUT ARGUMENTS:
0000 165
00000004 0000 166 SRC_STR = 4
00000008 0000 167 DST_STR = 8
0000 168
0000 169
40FC 0000 170 .ENTRY LIB$TRA_EBC_ASC, ^M<IV, R2, R3, R4, R5, R6, R7>
0002 171 ;+
0002 172 ; Extract the lengths and addresses we will need from the source and
0002 173 ; destination descriptors.
0002 174 ;--
50 04 AC DO 0002 175 MOVL SRC_STR(AP), R0 ; Address of src descriptor
02 03 AO 91 0006 176 CMPB DSC$B_CLASS(R0), #DSC$K_CLASS_D ; read like fixed ?
06 1A 000A 177 BGTRU 1$ ; no
56 04 BC 7D 000C 178 MOVQ @SRC_STR(AP), R6 ; length->R6, address->R7
09 11 0010 179 BRB 2$ ; join common flow
00000000'GF 16 0012 180
56 51 7D 0018 181 1$: JSB G^LIB$ANALYZE_SDESC_R2 ; Extract: length->R1, addr->R2
001B 182 MOVQ R1, R6 ; length->R6, address->R7
50 08 AC DO 001B 183
02 03 AO 91 001F 184 2$: MOVL DST_STR(AP), R0 ; Address of dst descriptor
06 1A 0023 185 CMPB DSC$B_CLASS(R0), #DSC$K_CLASS_D ; read like fixed ?
54 08 BC 7D 0025 186 BGTRU 3$ ; no
09 11 0029 187 MOVQ @DST_STR(AP), R4 ; length->R4, address->R5
002B 188 BRB 4$ ; join common flow
00000000'GF 16 002B 189
54 51 7D 0031 190 3$: JSB G^LIB$ANALYZE_SDESC_R2 ; Extract: length->R1, addr->R2
0034 191 MOVQ R1, R4 ; length->R4, address->R5
0034 192
0034 193 ;+
0034 194 ; If destination is a dynamic string, reallocate a dynamic string which
0034 195 ; is the same length as the input string.
0034 196 ;--
0034 197
52 08 AC DO 0034 198 4$: MOVL DST_STR(AP), R2 ; address of dst descriptor
02 03 A2 91 0038 199 CMPB DSC$B_CLASS(R2), #DSC$K_CLASS_D ; dynamic string dest ?
1C 12 003C 200 BNEQ 5$ ; not dynamic
7E 56 3C 003E 201 MOVZWL R6, -(SP) ; length of source
08 AC DD 0041 202 PUSHL DST_STR(AP) ; caller's dest desc addr
04 AE DF 0044 203 PUSHAL 4(SP) ; address of length of source
00000000'GF 02 FB 0047 204 CALLS #2, G^LIB$SGET1_DD ; allocate a dynamic string
004E 205 ; equal in len to source
5E 04 CO 004E 206 ADDL2 #4, SP ; restore stack
54 56 3C 0051 207 MOVZWL R6, R4 ; dest length = source length
55 04 A2 DO 0054 208 MOVL DSC$A_POINTER(R2), R5 ; new destination address
21 11 0058 209 BRB 10$ ; join common flow
005A 210 ;+
005A 211 ; If destination is a varying string, force its CURLEN to the MIN (
005A 212 ; MAXSTRLEN, length of source). If we can't, return LIB$_INVARG.
005A 213 ;--
005A 214 5$:
0B 03 A2 91 005A 215 CMPB DSC$B_CLASS(R2), #DSC$K_CLASS_VS ; varying string dest ?
0E 12 005E 216 BNEQ 6$ ; not varying
56 62 B1 0060 217 CMPW DSC$W_MAXSTRLEN(R2), R6 ; MAXSTRLEN ? length of source
0E 1F 0063 218 BLSSU 9$ ; won't fit, exit with error
54 56 3C 0065 219 MOVZWL R6, R4 ; Set dest len to source length
04 B2 54 B0 0068 220 MOVW R4, @DSC$A_POINTER(R2) ; Set CURLEN to new length
```

```
0D 11 006C 221 BRB 10$ ; join common flow
      006E 222 ;+
      006E 223 ; Otherwise, we have a destination string with fixed-length semantics.
      006E 224 ; Its length must match the length of the input string,
      006E 225 ; else LIB$_INVARG is returned.
      006E 226 ; -
      006E 227 6$:
54 56 B1 006E 228 CMPW R6, R4 ; source len = dest len ?
      08 13 0071 229 BEQL 10$ ; ok if lengths match
50 00000000'8F D0 0073 230 9$: MOVL #LIB$_INVARG, R0 ; return "invalid argument"
      04 007A 231 RET
      007B 232
      007B 233 ;+
      007B 234 ; Set up registers for the MOVTUC and MOVTC to follow
      007B 235 ; -
      007B 236 10$:
50 56 B0 007B 237 MOVW R6, R0 ; get source length in R0
51 57 D0 007E 238 MOVL R7, R1 ; address of caller's source
      0081 239 ;+
      0081 240 ; registers at this point...
      0081 241 ;
      0081 242 ; R0 length of source string
      0081 243 ; R1 address of source string
      0081 244 ; R2 address of destination descriptor
      0081 245 ; R3 unknown
      0081 246 ; R4 length of destination string (must be equal to R6)
      0081 247 ; R5 address of destination string
      0081 248 ; R6 length of source string (must be equal to R4)
      0081 249 ; R7 address of source string (same as R1)
      0081 250 ; -
```

```
00000000'GF 5C 8F 61 50 2F 0081 252 :+
                                0081 253 : Actual translation loop.
                                0081 254 : repeated until we translate all error free, or fall through with
                                0081 255 : an error and perform one final MOVTC to complete translation
                                0081 256 :-
                                0081 257
                                0081 258 15$:
00000000'GF 5C 8F 61 50 2F 0081 259      MOVTC R0, (R1), #ESC_CHAR, G^LIB$AB_EBC_ASC, R4, (R5)
                                008B 260
                                008D 261 : State of regs after a MOVTC instr.
                                008D 262 : R0 = number of bytes remaining in
                                008D 263 : source string (including the
                                008D 264 : byte which caused the escape.
                                008D 265 : Is zero only if the entire source
                                008D 266 : string was translated and moved
                                008D 267 : without escape.
                                008D 268 : R1 = address of the byte which
                                008D 269 : resulted in destination string
                                008D 270 : exhaustion or escape. If no
                                008D 271 : exhaustion or escape, then
                                008D 272 : address of one byte beyond the
                                008D 273 : source string.
                                008D 274 : R2 = 0
                                008D 275 : R3 = address of the table
                                008D 276 : R4 = number of bytes remaining in the
                                008D 277 : destinatin string.
                                008D 278 : R5 = address of the byte in the
                                008D 279 : destination string which would
                                008D 280 : have received the translated byte
                                008D 281 : that caused the escape or would
                                008D 282 : have received a translated byte
                                008D 283 : if the source string were not
                                008D 284 : exhausted. If not exhaustion
                                008D 285 : or escape, then address of one
                                008D 286 : byte beyond the destination
                                008D 287 : string.
                                008D 288      BVC GOOD_COMPL : terminated by end of string
                                008F 289      MOVBC #ESC_CHAR, (R5)+ : store esc char in output
                                0093 290 : bumping R5 to next byte pos.
                                0093 291      DECL R0 : adjust input count for one
                                0095 292 : done by hand
                                0095 293      DECL R4 : adjust output count for one
                                0097 294 : done by hand
                                0097 295      CMPB (R1)+, #SOURCE_CODE : was the input an escape char
                                009B 296      BEQL 15$ : yes -- treat as success and
                                009D 297 : and continue
                                009D 298
                                009D 299 :+
                                009D 300 : an untranslatable input char has been detected. Translate rest of
                                009D 301 : string and exit with an error status of LIB$INVCHA
                                009D 302 :-
                                009D 303
00000000'GF 6E 61 50 2E 009D 304      MOVTC R0, (R1), 0(SP), G^LIB$AB_EBC_ASC, R4, (R5)
                                00A6 305
                                00A8 306 : NOTE: lengths are guarenteed
                                00A8 306 : to be equal, so fill_char is
```

```
00A8 307 ; not needed.
00A8 308 ; NOTE: lengths are guaranteed
00A8 309 ; to be equal, so fill_char is
00A8 310 ; not needed.
00A8 311 ; State of regs after a MOVTC instr.
00A8 312 ; R0 = number of translated bytes
00A8 313 ; remaining in source string.
00A8 314 ; is non zero only if source string
00A8 315 ; is longer than destination
00A8 316 ; string.
00A8 317 ; R1 = address of one byte beyond the
00A8 318 ; last byte in source string that
00A8 319 ; was translated.
00A8 320 ; R2 = 0
00A8 321 ; R3 = address of the translation table
00A8 322 ; R4 = 0
00A8 323 ; R5 = address of one byte beyond the
00A8 324 ; destination string.
00A8 325 ;
50 00000000'8F D0 00A8 326 MOVL #LIB$_INVCHA, R0 ; return "invalid char" code
04 00AF 327 RET
00B0 328
00B0 329 GOOD_COMPL:
50 01 D0 00B0 330 MOVL #SS$_NORMAL, R0 ; return success
04 00B3 331 RET
00B4 332
00B4 333 .END
```

LIB\$TRA_EBC_ASC
Symbol Table

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 0?
LIB\$AB_EBC_ASC ***** X 00
LIB\$ANALYZE_SDESC_R2 ***** X 00
LIB\$GET1_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
\$ABSS	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=LISS:LIBTRAE2A/OBJ=OBJ\$:LIBTRAE2A MSRC\$:LIBTRAE2A/UPDATE=(ENH\$:LIBTRAE2A)

0210 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

