

[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

```

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
LLLLLLLLLLLL IIIIII          SSSSSSSS
LLLLLLLLLLLL IIIIII          SSSSSSSS

```


(2) 45
(3) 66
(5) 126

Edit History
DECLARATIONS
OTSS\$CNVOUT - Convert real to text

```
0000 1      .TITLE OTSSCNVOUT - Convert Real (D, G, H) to Text
0000 2      .IDENT /1-011/                               ; File: OTSCNVOUT.MAR EDIT:MDL1011
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 ++
0000 31 : FACILITY: Language Independent Support Library
0000 32
0000 33 : ABSTRACT:
0000 34
0000 35 :     Routines to convert D, G and H floating values to text representations.
0000 36 :     This module produces the FORTRAN E format as an aid
0000 37 :     to the conversion between D, F, Q, and Packed data types.
0000 38
0000 39 : ENVIRONMENT: User Mode, AST Reentrant
0000 40
0000 41 --
0000 42 : AUTHOR: R. REICHERT, CREATION DATE: 27-Sept-1979
0000 43 :
```



```

0000 45      .SBTTL Edit History
0000 46      :
0000 47      : 1-001 - Original by lifting code from FORCVTRT.MAR (Version 1-007)
0000 48      : Only the code in support of the entry point FOR$CNV_OUT_E
0000 49      : is reflected in this module. RKR 27-SEP-79
0000 50      : 1-002 - Cosmetic changes. RKR 21-OCT-79
0000 51      : 1-003 - Delete all code not needed by COBOL. RKR 7-NOV-79
0000 52      : 1-004 - Simplify code to find temp_string length (taken from FORCVTRT)
0000 53      : PDG 7-AUG-81
0000 54      : 1-005 - Added EDIT field for use in checkin's audit trail. Updated
0000 55      : copyright date. LB 24-AUG-81
0000 56      : 1-006 - Change source name to OTSS but retain old entry point name.
0000 57      : PLL 21-Jan-82
0000 58      : 1-007 - Add new entry points for g and h floating - previous version
0000 59      : did not call proper kernel conversion routines for g and h. PLL 8-Mar-82
0000 60      : 1-008 - Add COB$CNVOUT entry point. PLL 19-Mar-1982
0000 61      : 1-009 - COB$ entry point must branch around OTSS entry. PLL 19-Mar-1982
0000 62      : 1-010 - Make references to OTSS convert routines G^ to get rid of non-pic
0000 63      : problem. PLL 7-Jun-1982
0000 64      : 1-011 - Allow for 4-digit exponent (H-floating). MDL 15-Sep-1982

```

```
0000 66 .SBTTL DECLARATIONS
0000 67 :
0000 68 : INCLUDE FILES:
0000 69 :
0000 70 :
0000 71 :
0000 72 : EXTERNAL DECLARATIONS:
0000 73 :
0000 74 .DSABL GBL ; Prevent undeclared
0000 75 ; symbols from being
0000 76 ; automatically global.
0000 77 .EXTRN OTSS$CVT_D_T_R8 ; Kernel convert routine
0000 78 .EXTRN OTSS$CVT_G_T_R8 ; Kernel convert routine
0000 79 .EXTRN OTSS$CVT_H_T_R8 ; Kernel convert routine
0000 80 .EXTRN OTSS_FATINTERR ; Error code
0000 81
0000 82 :
0000 83 : MACROS:
0000 84 :
0000 85 :
0000 86 :
0000 87 : EQUATED SYMBOLS:
0000 88 :
0000 89 :
0000 90 ; Stack frame offsets from FP
0000 91 ;; Common frame for kernel convert routines
FFFFF8 0000 92 PACKED = -8 ; Temp for packed representation
FFFFF4 0000 93 FLAGS = PACKED - 4 ; Flags for outer and inner routines
FFFFF0 0000 94 SIG_DIGITS = FLAGS - 4 ; Significant digits
FFFFFEC 0000 95 STRING_ADDR = SIG_DIGITS - 4 ; Address of temp string
FFFFFE8 0000 96 SIGN = STRING_ADDR - 4 ; Sign
FFFFFE4 0000 97 DEC_EXP = SIGN - 4 ; Decimal exponent
FFFFFE0 0000 98 OFFSET = DEC_EXP - 4 ; Offset
FFFFFDC 0000 99 RT_RND = OFFSET - 4 ; Right round point
FFFFFDC 0000 100 COMMON_FRAME = RT_RND ; Common frame size
0000 101 ;; Not-in-common stack frame
FFFFFD8 0000 102 EXP_LETTER = COMMON_FRAME - 4 ; Exponent letter to use
FFFFFD4 0000 103 S_SCALE = EXP_LETTER - 4 ; Saved scale factor
FFFFFD0 0000 104 S_DI = S_SCALE - 4 ; Saved digits in integer
FFFFFCC 0000 105 S_DE = S_DI - 4 ; Saved digits in exponent
FFFFFC8 0000 106 S_DF = S_DE - 4 ; Saved digits in fraction
FFFFFC4 0000 107 LEAD_DIGITS = S_DF - 4 ; Number of leading digits
FFFFFC0 0000 108 LEAD_ZERO = LEAD_DIGITS - 4 ; Number of zeroes after decimal pt.
FFFFFBC 0000 109 TRAIL_DIGITS = LEAD_ZERO - 4 ; Number of trailing digits
FFFFFBC 0000 110 FRAME = TRAIL_DIGITS ; Frame size
0000 111
01000000 0000 112 M_TRUNCATE = 1a24 ; Flag to kernel routine
02000000 0000 113 M_RT_ROUND = 1a25 ;
0000 114
0000 115 :
0000 116 : OWN STORAGE:
0000 117 :
0000 118 :
0000 119 :
0000 120 : PSECT DECLARATIONS:
0000 121 :
00000000 122 .PSECT _OTSS$CODE PIC, USR, CON, REL, LCL, SHR, -
```

OTSSCNVOUT
1-011

- Convert Real (D, G, H) to Text I 10
DECLARATIONS

0000 123

16-SEP-1984 00:22:44
6-SEP-1984 11:12:45

VAX/VMS Macro V04-00
[LIBRTL.SRC]OTSSCNVOUT.MAR;1

Page 4
(3)

EXE, RD, NOWRT, LONG


```
0000 125      .SBTTL OTSS$CNVOUT - Convert real to text
0000 126
0000 127      ;++
0000 128      ; FUNCTIONAL DESCRIPTION:
0000 129
0000 130      ; This routine performs conversion of floating values
0000 131      ; to text representations.
0000 132
0000 133
0000 134      ; CALLING SEQUENCE:
0000 135      ; status.wlc.v = OTSS$CNVOUT (value.rx.r, out_string.wt.ds,
0000 136      ;                               digits_in_fract.rlu.v)
0000 137
0000 138
0000 139      ; INPUT PARAMETERS:
0000 140
00000004 0000 141      ; value = 4 ; The address of the floating value to be co
0000000C 0000 142      ; digits_in_fract = 12 ; The number of digits in the fraction
0000 143      ; portion.
0000 144
0000 145      ; IMPLICIT INPUTS:
0000 146
0000 147      ; NONE
0000 148
0000 149      ; OUTPUT PARAMETERS:
00000008 0000 150
0000 151      ; out_string = 8 ; The address of the descriptor of the resul
0000 152      ; string. The string must be
0000 153      ; class S. (Scalar)
0000 154
0000 155      ; IMPLICIT OUTPUTS:
0000 156
0000 157      ; NONE
0000 158
0000 159      ; COMPLETION CODES:
0000 160
0000 161      ; $$$ NORMAL - Successful completion
0000 162      ; OTSS$_FATINTERR - Output conversion error. The value did not
0000 163      ; fit in the given string.
0000 164
0000 165      ; SIDE EFFECTS:
0000 166
0000 167      ; $$$_ROPRAND - If the value to be converted is a reserved
0000 168      ; floating operand.
0000 169
0000 170      ;--
```



```
01FC 0000 172 .ENTRY COB$CNVOUT, ^M<R2,R3,R4,R5,R6,R7,R8>
02 11 0002 173 BRB SAME
01FC 0004 174 .ENTRY OTSS$CNVOUT, ^M<R2,R3,R4,R5,R6,R7,R8>
58 00000000'GF 9E 0006 175 SAME: MOVAB G^OTSS$CVT_D_T_R8, R8 ; addr of kernel convert routine
14 11 000D 176 BRB COMMON
000F 177
01FC 000F 178 .ENTRY OTSS$CNVOUT_G, ^M<R2,R3,R4,R5,R6,R7,R8>
58 00000000'GF 9E 0011 179 MOVAB G^OTSS$CVT_G_T_R8, R8 ; addr of kernel convert routine
09 11 0018 180 BRB COMMON
001A 181
01FC 001A 182 .ENTRY OTSS$CNVOUT_H, ^M<R2,R3,R4,R5,R6,R7,R8>
58 00000000'GF 9E 001C 183 MOVAB G^OTSS$CVT_H_T_R8, R8 ; addr of kernel convert routine
0023 184
0023 185 COMMON:
SE BC AD 9E 0023 186 MOVAB FRAME(FP), SP ; Set up stack frame
DB AD 45 8F 90 0027 187 MOVB #^A/E/, EXP_LETTER(FP) ; Use letter E
002C 188 ;+
002C 189 ; look at the argument list and fill in the appropriate
002C 190 ; values in the local frame.
002C 191 ; -
002C 192
002C 193 INITIALIZE:
CC AD F4 AD D4 002C 194 CLRL FLAGS(FP) ; No flags initially
DO AD D0 002F 195 MOVL #2, S_DE(FP) ; Digits in exponent
D4 AD D4 0033 196 CLRL S_DI(FP) ; Digits in integer
C8 AD 0C AC D0 0036 197 CLRL S_SCALE(FP) ; Scale factor
FO AD C8 AD D0 0039 198 90$: MOVL digits_in_fract(AP), S_DF(FP) ; Store digits_in_fract
52 FO AD 13 C1 003E 199 MOVL S_DF(FP), SIG_DIGITS(FP) ; Initial number of sig. digits
EC AD 5E 52 C2 0043 200 10$: ADDL3 #T9, SIG_DIGITS(FP), R2 ; Find temp_string length
50 AD 04 AC D0 0048 201 SUBL2 R2, SP ; Create string on stack
EC AD 5E 5E D0 004B 202 MOVL SP, STRING_ADDR(FP) ; Temp string address
51 5D D0 004F 203 MOVL value(AP), R0 ; Value address
EC AD E0 AD C0 0053 204 MOVL FP, R1 ; Local frame address
E8 AD D5 0056 205 JSB (R8) ; Call kernel conversion routine
03 12 0058 206 ADDL2 OFFSET(FP), STRING_ADDR(FP) ; Get first character pos.
BC AD C8 AD D0 005D 207 E_CONVERT:
C4 AD D4 0060 208 TSTL SIGN(FP) ; Is value zero?
C0 AD D4 0062 209 BNEQ 20$ ; No
E4 AD D4 0065 210 CLRL DEC_EXP(FP) ; Yes, exponent is zero
211 20$: MOVL S_DF(FP), TRAIL_DIGITS(FP)
212 CLRL LEAD_DIGITS(FP) ; Yes
213 CLRL LEAD_ZERO(FP)
214 30$:
215 ;+
216 ; Format the digits in the output string.
217 ;
218 ; The string will be constructed as follows:
219 ;
220 ; n1 blanks, where n1 is calculated
221 ; LEAD_DIGITS digits
222 ; a decimal point
223 ; LEAD_ZERO zeroes
224 ; TRAIL_DIGITS digits
225 ;
226 ; The sign is inserted where appropriate. If LEAD_DIGITS is
227 ; zero, an optional leading zero is inserted if there is
228 ; room.
```

```
0070 229 :-  
0070 230  
0070 231 FORM_DIGIT:  
56 EC AD D0 0070 232 MOVL STRING_ADDR(FP), R6 ; Address of first digit  
52 08 BC 7D 0074 233 MOVQ @out_string(AP), R2 ; Get string descriptor  
52 52 52 3C 0078 234 MOVZWL R2, R2  
50 52 C4 AD C3 007B 235 SUBL3 LEAD_DIGITS(FP), R2, R0 ; Find leading blanks  
50 50 BC AD C2 0080 236 SUBL2 TRAIL_DIGITS(FP), R0  
50 50 CC AD C2 0084 237 SUBL2 S_DE(FP), R0  
50 50 02 C2 0088 238 SUBL2 #2, R0 ; Compensate for exponent  
E8 AD D5 008B 239 10$: TSTL SIGN(FP) ; Negative?  
02 18 008E 240 BGEQ 20$ ; No  
50 D7 0090 241 15$: DECL R0 ; Use another character  
50 D7 0092 242 20$: DECL R0 ; For decimal point  
03 18 0094 243 BGEQ 24$ ; If room left  
009A 31 0096 244 BRW ERROR ; No room left  
57 50 D0 0099 245 24$: MOVL R0, R7  
06 13 009C 246 BEQL 22$ ; Skip if no blanks  
83 20 90 009E 247 21$: MOVB #^A/ /, (R3)+ ; Insert leading blanks  
FA 50 F5 00A1 248 SOBGR R0, 21$ ; Loop till done  
E8 AD D5 00A4 249 22$: TSTL SIGN(FP) ; Negative?  
03 18 00A7 250 BGEQ 30$ ; No  
83 2D 90 00A9 251 25$: MOVB #^A/- /, (R3)+ ; Minus sign  
00AC 252 30$: ;  
57 D5 00AC 253 35$: TSTL R7 ; Is there room?  
0E 15 00AE 254 BLEQ 40$ ; No, skip it  
20 73 91 00B0 255 37$: CMPB -(R3), #^A/ / ; Is last char a blank?  
04 13 00B3 256 BEQL 38$ ; Yes, dont move it  
FF A3 63 90 00B5 257 MOVB (R3), -1(R3) ; Move sign  
83 30 90 00B9 258 38$: MOVB #^A/0 /, (R3)+ ; Insert zero  
08 11 00BC 259 BRB 42$ ;  
63 66 C4 AD 28 00BE 260 40$: MOVC3 LEAD_DIGITS(FP), (R6), (R3) ; Move leading digits  
56 51 D0 00C3 261 MOVL R1, R6  
83 2E 90 00C6 262 42$: MOVB #^A/./, (R3)+ ; Move decimal point  
50 C0 AD D0 00C9 263 MOVL LEAD_ZERO(FP), R0 ; Insert leading zeroes  
06 15 00CD 264 BLEQ 50$ ; Skip if none  
83 30 90 00CF 265 45$: MOVB #^A/0 /, (R3)+ ; Move a zero  
FA 50 F5 00D2 266 SOBGR R0, 45$ ; Loop till done  
50 BC AD D0 00D5 267 50$: MOVL TRAIL_DIGITS(FP), R0 ; Move trailing digits  
04 15 00D9 268 BLEQ 60$ ; Skip if none  
63 66 50 28 00DB 269 MOVC3 R0, (R6), (R3) ; Move trailing digits  
00DF 270 60$: ;  
00DF 271 ;  
00DF 272 ;  
83 D8 AD 90 00DF 273 MOVB EXP_LETTER(FP), (R3)+ ; Move exponent letter  
54 53 D0 00E3 274 MOVL R3, R4 ; Save address  
64 F8 AD 05 E4 AD F9 00E6 275 CVTLP DEC_EXP(FP), #5, PACKED(FP) ; Convert exponent  
CC AD F8 AD 05 08 00EC 276 CVTPS #5, PACKED(FP), S_DE(FP), (R4) ; assuming its 2 digits long  
1F 1C 00F3 277 BVC 70$ ; Overflow?  
CC AD D6 00F5 278 INCL S_DE(FP) ; Try another digit  
53 D7 00F8 279 DECL R3  
63 CC AD F8 AD 05 08 00FA 280 CVTPS #5, PACKED(FP), S_DE(FP), (R3)  
11 1C 0101 281 BVC 70$ ; Overflow?  
CC AD D6 0103 282 INCL S_DE(FP) ; Try another digit  
53 D7 0106 283 DECL R3  
63 CC AD F8 AD 05 08 0108 284 CVTPS #5, PACKED(FP), S_DE(FP), (R3)  
03 1C 010F 285 BVC 70$ ; No overflow
```



```

53      001F 31 0111 286      BRW      ERROR      ; Overflow, must be >4 digits long
      CC AD  C0 0114 287 70$: ADDL2  S_DE(FP), R3      ; Move string pointer
      53 D6 0118 288
      011A 289 ;+
      011A 290 ; Blank fill the remainder of the string and return to the caller.
      011A 291 ; -
50      08 BC 7D 011A 292      MOVQ      @out_string(AP), R0      ; Get string descriptor
      011E 293
      011E 294 FINISH:
50      50 3C 011E 295      MOVZWL   R0, R0
51      50 C0 0121 296      ADDL2    R0, R1      ; Find last character
51      53 C2 0124 297      SUBL2    R3, R1      ; Subtract characters used
      06 13 0127 298      BEQL      EXIT      ; If none left, exit
63      51 20 6E 00 2C 0129 299      MOVCS   #0, (SP), #^A/ /, R1, (R3) ; Blank fill remainder
      50 01 D0 012F 300 EXIT:  MOVL     #1, R0      ; SSS_NORMAL
      04 0132 301      RET
      0133 302
      0133 303 ERROR:
61      50 2A 50 08 BC 7D 0133 304      MOVQ      @out_string(AP), R0
      50 6E 00 2C 0137 305      MOVCS   #0, (SP), #^A/*/, R0, (R1)
      00000000'8F D0 013D 306      MOVL     #OTSS_FATINTERR, R0
      04 0144 307      RET
      0145 308
      0145 309
      0145 310      .END
```

OTSSCNVOUT
Symbol table

- Convert Real (D, G, H) to Text N 10

16-SEP-1984 00:22:44 VAX/VMS Macro V04-00 Page 9
6-SEP-1984 11:12:45 [LIBRTL.SRC]OTSCNVOUT.MAR;1 (6)

```
COB$CNVOUT      00000000 RG    01
COMMON          00000023 R     01
COMMON FRAME    = FFFFFFFDC
DEC EXP         = FFFFFFFE4
DIGITS_IN_FRACT= 0000000C
ERROR           00000133 R     01
EXIT            0000012F R     01
EXP LETTER      = FFFFFFFD8
E_CONVERT       0000005D R     01
FINISH          0000011E R     01
FLAGS           = FFFFFFFF4
FORM DIGIT      00000070 R     01
FRAME           = FFFFFFFBC
INITIALIZE      0000002C R     01
LEAD_DIGITS     = FFFFFFFC4
LEAD_ZERO       = FFFFFFFC0
OFFSET          = FFFFFFFE0
OTSS$CVT_D-T-R8 ***** X    00
OTSS$CVT_G-T-R8 ***** X    00
OTSS$CVT_H-T-R8 ***** X    00
OTSSCNVOUT      00000004 RG    01
OTSSCNVOUT_G     0000000F RG    01
OTSSCNVOUT_H     0000001A RG    01
OTSS$FATINTERR ***** X    00
OUT STRING       = 00000008
PACKED           = FFFFFFFF8
RT_RND           = FFFFFFFDC
SAME             00000006 R     01
SIGN             = FFFFFFFE8
SIG DIGITS       = FFFFFFFF0
STRING_ADDR      = FFFFFFFEC
S_DE             = FFFFFFFCC
S_DF             = FFFFFFFC8
S_DI             = FFFFFFFD0
S_SCALE          = FFFFFFFD4
TRAIL_DIGITS     = FFFFFFFBC
VALUE            = 00000004
```

! Psect synopsis !

PSECT name	Allocation	PSECT No.	Attributes														
ABS	00000000 (0.)	00 (0.)	NOPIC	USR	CON	ABS	LCL	NOSHR	NOEXE	NORD	NOWRT	NOVEC	BYTE				
_OTSSCODE	00000145 (325.)	01 (1.)	PIC	USR	CON	REL	LCL	SHR	EXE	RD	NOWRT	NOVEC	LONG				

! Performance indicators !

Phase	Page faults	CPU Time	Elapsed Time
Initialization	32	00:00:00.07	00:00:00.90
Command processing	127	00:00:00.29	00:00:02.61
Pass 1	78	00:00:00.58	00:00:02.28
Symbol table sort	0	00:00:00.02	00:00:00.02

Pass 2	69	00:00:00.37	00:00:01.62
Symbol table output	4	00:00:00.03	00:00:00.03
Psect synopsis output	3	00:00:00.01	00:00:00.01
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	315	00:00:01.37	00:00:07.47

The working set limit was 900 pages.
5317 bytes (11 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 39 non-local and 21 local symbols.
310 source lines were read in Pass 1, producing 20 object records in Pass 2.
0 pages of virtual memory were used to define 0 macros.

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

Macro library name	Macros defined
-----	-----
_\$255\$DUA28:[SYSLIB]STARLET.MLB;2	0

0 GETS were required to define 0 macros.

There were no errors, warnings or information messages.

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

0211 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

