


```

FFFFFFFFF  000000  RRRRRRRR  CCCCCCCC  VV      VV  TTTTTTTTTT  RRRRRRRR  TTTTTTTTTT
FFFFFFFFF  000000  RRRRRRRR  CCCCCCCC  VV      VV  TTTTTTTTTT  RRRRRRRR  TTTTTTTTTT
FF          00      00  RR      RR  CC      CC      TT      RR      RR  TT
FF          00      00  RR      RR  CC      CC      TT      RR      RR  TT
FF          00      00  RR      RR  CC      CC      TT      RR      RR  TT
FF          00      00  RR      RR  CC      CC      TT      RR      RR  TT
FFFFFFFFF  00      00  RRRRRRRR  CC      CC      TT      RRRRRRRR  TT
FFFFFFFFF  00      00  RRRRRRRR  CC      CC      TT      RRRRRRRR  TT
FF          00      00  RR  RR  CC      CC      TT      RR  RR  TT
FF          00      00  RR  RR  CC      CC      TT      RR  RR  TT
FF          00      00  RR      RR  CC      CC      TT      RR      RR  TT
FF          00      00  RR      RR  CC      CC      TT      RR      RR  TT
FF          000000  RR      RR  CCCCCCCC  VV      VV  TT      RR      RR  TT
FF          000000  RR      RR  CCCCCCCC  VV      VV  TT      RR      RR  TT

```

```

LL          111111  SSSSSSSS
LL          111111  SSSSSSSS
LL          11      SS
LL          11      SS
LL          11      SS
LL          11      SS
LL          11      SSSSSS
LL          11      SSSSSS
LL          11      SS
LL          11      SS
LL          11      SS
LL          11      SS
LLLLLLLLLLL 111111  SSSSSSSS
LLLLLLLLLLL 111111  SSSSSSSS

```

(2)	43	Edit History
(3)	70	DECLARATIONS
(4)	132	FOR\$CVT_x_Ty - Convert real to text
(5)	317	FOR\$CVT_x_TF - Fixed point format
(6)	379	FOR\$CVT_x_TG - Convert real to text - G format
(7)	485	INITIALIZE - Analyze argument list
(8)	517	INITIALIZE_F - Analyze argument list for F format
(9)	541	DIGITS_OUT
(10)	618	FINISH

```

0000 1      .TITLE FOR$CVTRT - Convert Real (F, D, G, H) to Text
0000 2      .IDENT /1-017/ ; File: FORCVTRT.MAR Edit: SBL1017
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: FORTRAN Language Support Library
0000 31 :
0000 32 : ABSTRACT:
0000 33 :
0000 34 :     Routines to convert F, D, G and H floating values to text representations.
0000 35 :     This module supports FORTRAN D, E, F and G formatting.
0000 36 :
0000 37 : ENVIRONMENT: User Mode, AST Reentrant
0000 38 :
0000 39 :--
0000 40 : AUTHOR: Steven B. Lionel, CREATION DATE: 19-Jun-1979
0000 41 :

```

```
0000 43      .SBTTL Edit History
0000 44      :
0000 45      : 1-001 - Original. SBL 19-Jun-1979
0000 46      : 1-002 - Remove STRING_LEN from stack frame. SBL 11-Jul-1979
0000 47      : 1-003 - Fix bug in rounding. SBL 6-Jul-1979
0000 48      : 1-004 - Keep sign on F underflow to zero. SBL 16-July-1979
0000 49      : 1-005 - Fix bug in G format rounding with digits in int. SBL 25-July-1979
0000 50      : 1-007 - Negate 1-004, that is not the desired behavior. SBL 20-Aug-1979
0000 51      : 1-008 - Speed optimizations. Use CASE for argument pickup. Use
0000 52      : fast way of computing temp string length. SBL 27-Dec-1979
0000 53      : 1-009 - Use correct limit in CASE statements. SBL 28-Dec-1979
0000 54      : 1-010 - Use signed branch on CASEB. SBL 31-Dec-1979
0000 55      : 1-011 - Do correct thing for >128 arguments. SBL 31-Dec-1979
0000 56      : 1-012 - Allow the digits-in-fract parameter to be optional, as in V1.
0000 57      : SBL 11-June-1980
0000 58      : 1-013 - Fix bug in 1-012 where FLAGS wasn't always cleared correctly.
0000 59      : SBL 27-June-1980
0000 60      : 1-014 - If V_ERR_OFLO is set, indicating Ew.dEe or Gw.dEe editing,
0000 61      : return error if exponent overflows (don't drop the letter E).
0000 62      : SPR 11-38351. JAW 15-Jun-1981
0000 63      : 1-015 - Return error if F or G-format width is too small. SPR 11-46071
0000 64      : SBL 4-May-1982
0000 65      : 1-016 - Don't force leading zero if there are any other digits at all
0000 66      : in the output. SBL 6-Oct-1982
0000 67      : 1-017 - Use general addressing mode. Add F routines. SBL 29-Oct-1982
0000 68
```

```

0000 70      .SBTTL  DECLARATIONS
0000 71      ;
0000 72      ; INCLUDE FILES:
0000 73      ;
0000 74      ;
0000 75      ;
0000 76      ; EXTERNAL DECLARATIONS:
0000 77      ;
0000 78      .DSABL  GBL                ; Prevent undeclared
0000 79      ;                          ; symbols from being
0000 80      ;                          ; automatically global.
0000 81      .EXTRN  OT$$$CVT_F_T_R8    ; Kernel convert routine
0000 82      .EXTRN  OT$$$CVT_D_T_R8    ; For D floating
0000 83      .EXTRN  OT$$$CVT_G_T_R8    ; For G floating
0000 84      .EXTRN  OT$$$CVT_H_T_R8    ; For H floating
0000 85      .EXTRN  FOR$_OUTCONERR      ; Error code
0000 86      ;
0000 87      ;
0000 88      ; MACROS:
0000 89      ;
0000 90      ;
0000 91      ;
0000 92      ; EQUATED SYMBOLS:
0000 93      ;
000001FC 0000 94      REGMASK = ^M<R2, R3, R4, R5, R6, R7, R8>
0000 95      ;
0000 96      ; Stack frame offsets from FP
0000 97      ;: Common frame for kernel convert routines
FFFFFFFF8 0000 98      PACKED = -8                ; Temp for packed representation
FFFFFFFF4 0000 99      FLAGS = PACKED - 4          ; Flags for outer and inner routines
FFFFFFFF0 0000 100     SIG_DIGITS = FLAGS - 4       ; Significant digits
FFFFFFFEC 0000 101     STRING_ADDR = SIG_DIGITS - 4 ; Address of temp string
FFFFFFFE8 0000 102     SIGN = STRING_ADDR - 4      ; Sign
FFFFFFFE4 0000 103     DEC_EXP = SIGN - 4          ; Decimal exponent
FFFFFFFE0 0000 104     OFFSET = DEC_EXP - 4       ; Offset
FFFFFFFDC 0000 105     RT_RND = OFFSET - 4        ; Right round point
FFFFFFFDC 0000 106     COMMON_FRAME = RT_RND      ; Common frame size
0000 107     ;: Not-in-common stack frame
FFFFFFFDB 0000 108     EXP_LETTER = COMMON_FRAME - 4 ; Exponent letter to use
FFFFFFFD4 0000 109     S_SCALE = EXP_LETTER - 4    ; Saved scale factor
FFFFFFFD0 0000 110     S_DI = S_SCALE - 4         ; Saved digits in integer
FFFFFFFCC 0000 111     S_DE = S_DI - 4           ; Saved digits in exponent
FFFFFFFC8 0000 112     S_DF = S_DE - 4           ; Saved digits in fraction
FFFFFFFC4 0000 113     LEAD_DIGITS = S_DF - 4     ; Number of leading digits
FFFFFFFC0 0000 114     LEAD_ZERO = LEAD_DIGITS - 4 ; Number of zeroes after decimal pt.
FFFFFFFBC 0000 115     TRAIL_DIGITS = LEAD_ZERO - 4 ; Number of trailing digits
FFFFFFFBC 0000 116     FRAME = TRAIL_DIGITS      ; Frame size
0000 117     ;
01000000 0000 118     M_TRUNCATE = 1a24          ; Flag to kernel routine
02000000 0000 119     M_RT_ROUND = 1a25          ;
0000 120     ;
0000 121     ;
0000 122     ; OWN STORAGE:
0000 123     ;
0000 124     ;
0000 125     ;
0000 126     ; PSECT DECLARATIONS:

```

FOR\$CVTRT
1-017

- Convert Real (F, D, G, H) to Text^{E 11}
DECLARATIONS

15-SEP-1984 23:49:35 VAX/VMS Macro V04-00
6-SEP-1984 10:54:25 [FORRTL.SRC]FOR\$CVTRT.MAR;1

Page 4
(3)

0000 127 ;
00000000 128
0000 129
0000 130

.PSECT _FOR\$CODE PIC, USR, CON, REL, LCL, SHR, -
EXE, RD, NOWRT, LONG

```

0000 132 .SBTTL FOR$CVT_x_Ty - Convert real to text
0000 133 :++
0000 134 : FUNCTIONAL DESCRIPTION:
0000 135 :
0000 136 : These routines perform conversion of floating values
0000 137 : to text representations. They are divided according to the
0000 138 : FORTRAN format types and by data type.
0000 139 :
0000 140 : The FORTRAN format types are D/E (exponential),
0000 141 : F (fixed point) and G (fixed or exponential).
0000 142 :
0000 143 : CALLING SEQUENCE:
0000 144 : status.wlc.v = FOR$CVT_x_Ty (value.rx.r, out_string.wt.ds
0000 145 : [ , digits_in_fract.rlu.v
0000 146 : [ , scale_factor.rl.v
0000 147 : [ , digits_in_int.rlu.v
0000 148 : [ , digits_in_exp.rlu.v
0000 149 : [ , caller_flags.rlu.v]]])
0000 150 :
0000 151 : where:
0000 152 : x is the datatype, either F, D, G or H
0000 153 : y is the format, one of D, E, F and G
0000 154 :
0000 155 : INPUT PARAMETERS:
0000 156 :
00000004 0000 157 : value = 4 ; The address of the floating value to be co
0000000C 0000 158 : digits_in_fract = 12 ; The number of digits in the fraction
00000010 0000 159 : scale_factor = 16 ; portion. Optional, assumed 0.
0000 160 : Optional, assumed 0. If
0000 161 : digits_in_int is not present,
0000 162 : scale_factor takes on the
0000 163 : FORTRAN semantics, i.e. indicating
0000 164 : the true scale factor on F format
0000 165 : or the digits_in_int for D, E and G
0000 166 : formats. The scale factor effect
0000 167 : is that the externally represented
0000 168 : number equals the internally
0000 169 : represented number multiplied
0000 170 : by 10**scale_factor.
00000014 0000 171 : digits_in_int = 20 ; The number of digits in the
0000 172 : integer part of an exponentially
0000 173 : formatted value. Optional,
0000 174 : assumed 0. Ignored for F
0000 175 : format.
00000018 0000 176 : digits_in_exp = 24 ; The number of digits in the
0000 177 : exponent field. Optional,
0000 178 : assumed 2. If the exponent
0000 179 : overflows this field by 1 digit,
0000 180 : the exponent letter is removed.
0000001C 0000 181 : caller_flags = 28 ; Optional, assumed 0.
00000000 0000 182 : V_FORCEPLUS = 0 ; Force + on positive values
00000001 0000 183 : V_ERR_OFLO = 1 ; Exponent field width overflow
0000 184 : is an error.
0000 185 :
0000 186 : IMPLICIT INPUTS:
0000 187 :
0000 188 : NONE

```



```

0000 189 :
0000 190 : OUTPUT PARAMETERS:
0000 191 :
00000008 0000 192 out_string = 8 ; The address of the descriptor of the resul
0000 193 ; string. The string must be
0000 194 ; class S. (Scalar)
0000 195 :
0000 196 : IMPLICIT OUTPUTS:
0000 197 :
0000 198 NONE
0000 199 :
0000 200 : COMPLETION CODES:
0000 201 :
0000 202 $$$ NORMAL - Successful completion
0000 203 FOR$_OUTCONERR - Output conversion error. The value did not
0000 204 fit in the given string.
0000 205 :
0000 206 : SIDE EFFECTS:
0000 207 :
0000 208 $$$_ROPRAND - If the value to be converted is a reserved
0000 209 floating operand.
0000 210 :
0000 211 :--
0000 212 :
01FC 0000 213 .ENTRY FOR$CVT_F_TE, REGMASK
58 D8 AD 5E BC AD 9E 0002 214 MOVAB FRAME(FP), SP ; Set up stack frame
00000000'GF 9E 0006 215 MOVB #^A/E/, EXP_LETTER(FP) ; Use letter E
008C 31 000B 216 MOVAB G^OT$$$CVT_F_T_RB, R8 ; Convert routine address
0012 217 BRW COMMON_E
0015 218
0015 219 FOR$CNV_OUT_E::
01FC 0015 220 .ENTRY FOR$CVT_D_TE, REGMASK
58 D8 AD 5E BC AD 9E 0017 221 MOVAB FRAME(FP), SP ; Set up stack frame
00000000'GF 9E 001B 222 MOVB #^A/E/, EXP_LETTER(FP) ; Use letter E
78 11 0020 223 MOVAB G^OT$$$CVT_D_T_RB, R8 ; Convert routine address
0027 224 BRB COMMON_E
0029 225
01FC 0029 226 .ENTRY FOR$CVT_G_TE, REGMASK
58 D8 AD 5E BC AD 9E 002B 227 MOVAB FRAME(FP), SP ; Set up stack frame
00000000'GF 9E C02F 228 MOVB #^A/E/, EXP_LETTER(FP) ; Use letter E
64 11 0034 229 MOVAB G^OT$$$CVT_G_T_RB, R8 ; Convert routine address
003B 230 BRB COMMON_E
003D 231
01FC 003D 232 .ENTRY FOR$CVT_H_TE, REGMASK
58 D8 AD 5E BC AD 9E 003F 233 MOVAB FRAME(FP), SP ; Set up stack frame
00000000'GF 9E 0043 234 MOVB #^A/E/, EXP_LETTER(FP) ; Use letter E
50 11 0048 235 MOVAB G^OT$$$CVT_H_T_RB, R8 ; Convert routine address
004F 236 BRB COMMON_E
0051 237
01FC 0051 238 .ENTRY FOR$CVT_F_TD, REGMASK
58 D8 AD 5E BC AD 9E 0053 239 MOVAB FRAME(FP), SP ; Set up stack frame
00000000'GF 9E 0057 240 MOVB #^A/D/, EXP_LETTER(FP) ; Use letter D
3C 11 005C 241 MOVAB G^OT$$$CVT_F_T_RB, R8 ; Convert routine address
0063 242 BRB COMMON_E
0065 243
01FC 0065 244 FOR$CNV_OUT_D::
0065 245 .ENTRY FOR$CVT_D_TD, REGMASK

```

```

58  SE BC AD 9E 0067 246 MOVAB FRAME(FP), SP ; Set up stack frame
    DB AD 44 8F 90 006B 247 MOVAB #^A/D/, EXP_LETTER(FP) ; Use letter D
    00000000'GF 9E 0070 248 MOVAB G^OTSS$CVT_D_T_RB, RB ; Convert routine address
    28 11 0077 249 BRB COMMON_E
    0079 250
    01FC 0079 251 .ENTRY FOR$CVT_G_TD, REGMASK
    SE BC AD 9E 007B 252 MOVAB FRAME(FP), SP ; Set up stack frame
    DB AD 44 8F 90 007F 253 MOVAB #^A/D/, EXP_LETTER(FP) ; Use letter D
    00000000'GF 9E 0084 254 MOVAB G^OTSS$CVT_G_T_RB, RB ; Convert routine address
    14 11 008B 255 BRB COMMON_E
    008D 256
    01FC 008D 257 .ENTRY FOR$CVT_H_TD, REGMASK
    SE BC AD 9E 008F 258 MOVAB FRAME(FP), SP ; Set up stack frame
    DB AD 44 8F 90 0093 259 MOVAB #^A/D/, EXP_LETTER(FP) ; Use letter D
    00000000'GF 9E 0098 260 MOVAB G^OTSS$CVT_H_T_RB, RB ; Convert routine address
    00 11 009F 261 BRB COMMON_E
    00A1 262
    00A1 263 COMMON_E:
    FO AD 029B 30 00A1 264 BSBW INITIALIZE ; Analyze argument list
    CB AD D0 00A4 265 MOVL S_DF(FP), SIG_DIGITS(FP) ; Initial number of sig. digits
    DO AD D5 00A9 266 TSTL S_DI(FP) ; Digits_in_int >=0?
    OF 13 00AC 267 BEQL 10$ ; Equal to zero
    OA 14 00AE 268 BGTR 5$ ; Greater than zero
    FO AD D0 AD C0 00B0 269 ADDL2 S_DI(FP), SIG_DIGITS(FP)
    06 14 00B5 270 BGTR 10$ ; Must be positive
    038D 31 00B7 271 BRW ERROR ; No significant digits!
    FO AD D6 00BA 272 5$: INCL SIG_DIGITS(FP) ; Get 1 more significant digit
    13 C1 00BD 273 10$: ADDL3 #19, SIG_DIGITS(FP), R2 ; Find temp_string length
    SE 52 C2 00C2 274 SUBL2 R2, SP ; Create string on stack
    EC AD 5E D0 00C5 275 MOVL SP, STRING_ADDR(FP) ; Temp string address
    50 04 AC D0 00C9 276 MOVL value(AP), -R0 ; Value address
    51 5D D0 00CD 277 MOVL FP, R1 ; Local frame address
    68 16 00D0 278 JSB (R8) ; Call kernel conversion routine
    EC AD E0 AD C0 00D2 279 ADDL2 OFFSET(FP), STRING_ADDR(FP) ; Get first character pos.
    E4 AD D4 AD C0 00D7 280 ADDL2 S_SCALE(FP), DEC_EXP(FP) ; Adjust exponent for scale
    00DC 281 E_CONVERT:
    E4 AD D0 AD C2 00DC 282 SUBL2 S_DI(FP), DEC_EXP(FP) ; Adjust for digits in int
    E8 AD D5 00E1 283 TSTL SIGN(FP) ; Is value zero?
    03 12 00F4 284 BNEQ 20$ ; No
    E4 AD D4 00E6 285 CLRL DEC_EXP(FP) ; Yes, exponent is zero
    BC AD C8 AD D0 00E9 286 20$: MOVL S_DF(FP), TRAIL_DIGITS(FP)
    C4 AD D0 AD D0 00EE 287 MOVL S_DI(FP), LEAD_DIGITS(FP) ; Number of leading digits
    11 14 00F3 288 BGTR 25$ ; Negative?
    C4 AD D4 00F5 289 CLRL LEAD_DIGITS(FP) ; Yes
    CO AD D0 AD CE 00F8 290 MNEGL S_DI(FP), LEAD_ZERO(FP) ; Number of zeroes after dec. pt.
    BC AD D0 AD C0 00FD 291 ADDL2 S_DI(FP), TRAIL_DIGITS(FP)
    12 14 0102 292 BGTR 30$ ; Must be positive
    OD 11 0104 293 BRB 29$ ; Otherwise error
    BC AD CO AD D4 0106 294 25$: CLRL LEAD_ZERO(FP) ; No leading zeroes
    DO AD C2 0109 295 SUBL2 S_DI(FP), TRAIL_DIGITS(FP)
    BC AD D6 010E 296 INCL TRAIL_DIGITS(FP)
    03 18 0111 297 BGEQ 30$ ; Can't be negative
    0331 31 0113 298 29$: BRW ERROR
    0294 30 0116 299 30$: BSBW DIGITS_OUT ; Output digits
    83 DB AD 90 0119 300 MOVAB EXP_LETTER(FP), (R3)+ ; Move exponent letter
    54 53 D0 011D 301 MOVL R3, -R4 ; Save address
    F8 AD 05 E4 AD F9 0120 302 CVTLP DEC_EXP(FP), #5, PACKED(FP) ; Convert exponent

```

64	CC AD	F8 AD	05	08	0126	303		CVTPS	#5, PACKED(FP), S_DE(FP), (R4)	
			16	1C	0 2D	304		BVC	35\$; Overflow?
	OE F4 AD		01	E0	012F	305		BBS	#V_ERR_OFLO, FLAGS(FP), 34\$; Yes: if exponent field
					0134	306				; width overflow is an error,
					0134	307				; don't drop the E.
		CC AD	D6		0134	308		INCL	S_DE(FP)	; Try another digit
		53	D7		0137	309		DECL	R3	
63	CC AD	F8 AD	05	08	0139	310		CVTPS	#5, PACKED(FP), S_DE(FP), (R3)	
			03	1C	0140	311		BVC	35\$; No overflow
		0302	31		0142	312	34\$:	BRW	ERROR	; Overflow
	53	CC AD	C0		0145	313	35\$:	ADDL2	S_DE(FP), R3	; Move string pointer
		53	D6		0149	314		INCL	R3	
	030B		31		014B	315		BRW	FINISH	; Finish up string

```

014E 317 .SBTTL FOR$CVT_x_TF - Fixed point format
014E 318
58 SE BC AD 01FC 014E 319 .ENTRY FOR$CVT_F_TF, REGMASK
00000000'GF 9E 0150 320 MOVAB FRAME(FP), SP ; Set up stack frame
2D 11 0154 321 MOVAB G^OTSS$CVT_F_T_R8, R8 ; Convert routine address
015B 322 BRB COMMON_F
015D 323
015D 324 FOR$CNV_OUT_F::
01FC 015D 325 .ENTRY FOR$CVT_D_TF, REGMASK
58 SE BC AD 9E 015F 326 MOVAB FRAME(FP), SP ; Set up stack frame
00000000'GF 9E 0163 327 MOVAB G^OTSS$CVT_D_T_R8, R8 ; Convert routine address
1E 11 016A 328 BRB COMMON_F
016C 329
01FC 016C 330 .ENTRY FOR$CVT_G_TF, REGMASK
58 SE BC AD 9E 016E 331 MOVAB FRAME(FP), SP ; Set up stack frame
00000000'GF 9E 0172 332 MOVAB G^OTSS$CVT_G_T_R8, R8 ; Convert routine address
OF 11 0179 333 BRB COMMON_F
017B 334
01FC 017B 335 .ENTRY FOR$CVT_H_TF, REGMASK
58 SE BC AD 9E 017D 336 MOVAB FRAME(FP), SP ; Set up stack frame
00000000'GF 9E 0181 337 MOVAB G^OTSS$CVT_H_T_R8, R8 ; Convert routine address
00 11 0188 338 BRB COMMON_F
018A 339
018A 340 COMMON_F:
01FC 30 018A 341 BSBW INITIALIZE F ; Analyze argument list
DB AD 94 018D 342 CLRB EXP LETTER(FP) ; Indicates F format
FO AD 08 BC 3C 0190 343 MOVZWL @out_string(AP), SIG_DIGITS(FP) ; Get field width
FO AD D7 0195 344 DECL SIG_DIGITS(FP) ; Max number of sig. digits
69 15 0198 345 BLEQ FG_ERROR ; No significant digits?
52 FO AD 13 C1 019A 346 ADDL3 #19, SIG_DIGITS(FP), R2 ; Calculate temp string size
SE 52 C2 019F 347 SUBL2 R2, SP ; Create string on stack
EC AD 5E D0 01A2 348 MOVL SP, STRING_ADDR(FP) ; String address
F4 AD 02000000 8F C8 01A6 349 BISL #M_RT_ROUND, FLAGS(FP) ; Flag indicating right round
DC AD D4 AD C8 AD C1 01AE 350 ADDL3 S_DF(FP), S_SCALE(FP), RT_RND(FP) ; Rounding position
50 04 AC D0 01B5 351 MOVL value(AP), R0 ; Value address
51 5D D0 01B9 352 MOVL FP, R1 ; Local frame pointer
68 16 01BC 353 JSB (R8) ; Do the conversion
EC AD E0 AD C0 01BE 354 ADDL2 OFFSET(FP), STRING_ADDR(FP) ; Get first digit pos.
E4 AD D4 AD C0 01C3 355 ADDL2 S_SCALE(FP), DEC_EXP(FP) ; Adjust for scale factor
01C8 356 F_CONVERT:
E8 AD D5 01C8 357 TSTL SIGN(FP) ; Is value zero?
03 12 01CB 358 BNEQ 10$ ; If zero
C4 AD E4 AD D4 01CD 359 CLRL DEC_EXP(FP) ; Then exponent is zero
E4 AD D0 01D0 360 10$: MOVL DEC_EXP(FP), LEAD_DIGITS(FP) ; Number of leading digits
03 18 01D5 361 BGEQ 20$ ; If greater than 0
C4 AD D4 01D7 362 CLRL LEAD_DIGITS(FP) ; Else no leading digits
CO AD E4 AD CE 01DA 363 20$: MNEGL DEC_EXP(FP), LEAD_ZERO(FP) ; Number of zeroes after dec pt.
03 18 01DF 364 BGEQ 30$ ; If greater than 0
CO AD C0 AD D4 01E1 365 CLRL LEAD_ZERO(FP) ; Else no leading zeroes
BC AD C8 AD C3 01E4 366 30$: SUBL3 LEAD_ZERO(FP), S_DF(FP), TRAIL_DIGITS(FP)
10 14 01EB 367 BGTR 35$ ; If not positive
CO AD BC AD D4 01ED 368 CLRL TRAIL_DIGITS(FP) ; Then no trailing digits
C8 AD D0 01F0 369 MOVL S_DF(FP), LEAD_ZERO(FP)
C4 AD D5 01F5 370 TSTL LEAD_DIGITS(FP) ; Any significant digits?
03 14 01F8 371 BGTR 35$ ; Yes
E8 AD D4 01FA 372 CLRL SIGN(FP) ; No, value is +0
01AD 30 01FD 373 35$: BSBW DIGITS_OUT ; Format the digits

```

FOR\$CVTRT
1-017

- Convert Real (F, D, G, H) to Text ^{K 11}
FOR\$CVT_x_TF - Fixed point format

15-SEP-1984 23:49:35 VAX/VMS Macro V04-00
6-SEP-1984 10:54:25 [FORRTL.SRC]FOR\$CVTRT.MAR;1

Page 10
(5)

0256	31	0200	374	BRW	FINISH	; Clean up and exit
		0203	375			
		0203	376	FG_ERROR:		
0241	31	0203	377	BRW	ERROR	; Return conversion error

```

0206 379 .SBTTL FOR$CVT_x_TG - Convert real to text - G format
0206 380
01FC 0206 381 .ENTRY FOR$CVT_F_TG, REGMASK
58 DB AD SE BC AD 9E 0208 382 MOVAB FRAME(FP), SP ; Set up stack frame
00000000'GF 90 020C 383 MOVB #*A/E/, EXP LETTER(FP) ; Use letter E for E format
3C 11 0211 384 MOVAB G^OTSS$CVT_F_T_R8, R8 ; Convert routine address
0218 385 BRB COMMON_G
021A 386
021A 387 FOR$CNV_OUT_G::
01FC 021A 388 .ENTRY FOR$CVT_D_TG, REGMASK
58 DB AD SE BC AD 9E 021C 389 MOVAB FRAME(FP), SP ; Set up stack frame
00000000'GF 90 0220 390 MOVB #*A/E/, EXP LETTER(FP) ; Use letter E for E format
28 11 0225 391 MOVAB G^OTSS$CVT_D_T_R8, R8 ; Convert routine address
022C 392 BRB COMMON_G
022E 393
022E 394 .ENTRY FOR$CVT_G_TG, REGMASK
58 DB AD SE BC AD 9E 0230 395 MOVAB FRAME(FP), SP ; Set up stack frame
00000000'GF 90 0234 396 MOVB #*A/E/, EXP LETTER(FP) ; Use letter E for E format
14 11 0239 397 MOVAB G^OTSS$CVT_G_T_R8, R8 ; Convert routine address
0240 398 BRB COMMON_G
0242 399
0242 400 .ENTRY FOR$CVT_H_TG, REGMASK
58 DB AD SE BC AD 9E 0244 401 MOVAB FRAME(FP), SP ; Set up stack frame
00000000'GF 90 0248 402 MOVB #*A/E/, EXP LETTER(FP) ; Use letter E for E format
00 11 024D 403 MOVAB G^OTSS$CVT_H_T_R8, R8 ; Convert routine address
0254 404 BRB COMMON_G
0256 405
0256 406 COMMON_G:
50 FO AD 00E6 30 0256 407 BSBW INITIALIZE ; Analyze argument list
CC AD 08 BC 3C 0259 408 MOVZWL @out string(AP), SIG_DIGITS(FP) ; Initial number
FO AD 50 C1 025E 409 ADDL3 #2, S DE(FP), R0 ; Get number needed
9A 15 0263 410 SUBL2 R0, SIG_DIGITS(FP)
52 FO AD 9A 15 0267 411 BLEQ FG ERROR ; No significant digits?
5E 52 C1 0269 412 ADDL3 #19, SIG_DIGITS(FP), R2 ; Calculate temp string size
EC AD 5E 52 C2 026E 413 SUBL2 R2, SP ; Create string on stack
F4 AD 01000000 8F C8 0271 414 MOVL SP, STRING_ADDR(FP) ; String address
50 04 AC D0 0275 415 BISL #M TRUNCATE, FLAGS(FP) ; Don't round
51 5D D0 027D 416 MOVL vaTue(AP), R0 ; Value address
68 16 0281 417 MOVL FP, R1 ; Local frame pointer
EC AD E0 AD C0 0284 418 JSB (R8) ; Do the conversion
54 EC AD D0 0286 419 ADDL2 OFFSET(FP), STRING_ADDR(FP) ; Get first digit pos.
E8 AD D5 028B 420 MOVL STRING_ADDR(FP), R4 ; R4 points to it
63 13 028F 421 TSTL SIGN(FP) ; Zero?
E4 AD D4 AD C0 0292 422 BEQL USE E ; Use E format
31 18 0294 423 ADDL2 S SCALE(FP), DEC_EXP(FP) ; Adjust for true scale factor
FFFFFFFF 8F E4 AD D1 0299 424 BGEQ TRY F ; Value >= 0.1, try F conversion
51 C8 AD D0 AD C1 029B 425 CMPL DEC_EXP(FP), #-1 ; Is it less than 0.1?
64 51 39 6E 00 2D 02A3 426 BLSS USE E ; Yes, use E conversion
35 63 91 02A5 427 ADDL3 S DI(FP), S DF(FP), R1 ; Will it round to 0.1?
3F 19 02AB 428 CMPC5 #0, (SP), #*A/9/, R1, (R4) ; .099999...?
64 FO AD 30 6E 00 2C 02B1 429 BNEQ USE E ; No, use E format
EC AD D7 02B3 430 CMPB (R3), #*A/5/ ; will it round?
E4 AD D6 02B6 431 BLSS USE E ; No, use E format
A4 31 90 02B8 432 MOVB #*A71/, -1(R4) ; Change to 0.10000
64 FO AD 30 6E 00 2C 02BC 433 MOVCS #0, (SP), #*A/0/, SIG_DIGITS(FP), (R4)
EC AD D7 02C3 434 DECL STRING_ADDR(FP) ; Value is now 0.10000!
E4 AD D6 02C6 435 INCL DEC_EXP(FP)

```

```

FEFC 31 02C9 436 BRW F_CONVERT ; Go to F conversion section
                                437 TRY_F:
CB AD E4 AD D1 02CC 438 CMPL DEC_EXP(FP), S_DF(FP) ; Too big for F?
    24 14 02D1 439 BGTR USE_E ; Yes, use E format
    3A 19 02D3 440 BLSS USE_F ; No, use F
64 E4 AD 39 6E 00 2D 02D5 441 CMPC5 #0, (SP), #^A/9/, DEC_EXP(FP), (R4) ; 9999...?
    31 12 02DC 442 BNEQ USE_F ; No, use F format
    35 63 91 02DE 443 CMPB (R3), #^A/5/ ; 9999....9995?
    2C 19 02E1 444 BLSS USE_F ; Won't round, use it as is
64 FO AD 30 FF A4 31 90 02E3 445 MOVB #^A71/, -1(R4) ; Change to 10000....
    6E 00 2C 02E7 446 MOVCS #0, (SP), #^A/0/, SIG_DIGITS(FP), (R4)
    EC AD D7 02EE 447 DECL STRING_ADDR(FP) ; Now changed
    E4 AD D6 02F1 448 INCL DEC_EXP(FP)
    FDE5 31 02F4 449 BRW E_CONVERT ; Go to E conversion routine
    50 CB AD D0 02F7 450 USE_E:
    DO AD D5 02FB 451 MOVL S_DF(FP), R0 ; Initial rounding position
    0A 13 02FE 452 TSTL S_DI(FP) ; In which direction to move?
    04 19 0300 453 BEQL 20$ ; =0, Nowhere
    50 D6 0302 454 BLSS 10$ ; < 0?
    04 11 0304 455 INCL R0 ; F+1 digits
    50 DO AD C0 0306 457 10$: ADDL2 S_DI(FP), R0 ; F-1: digits
    11 10 030A 458 20$: BSBB G_ROUND ; round the value
    FDCD 31 030C 459 BRW E_CONVERT ; And finish with E conversion
    50 CB AD D0 030F 460 USE_F:
    0B 10 0313 461 MOVL S_DF(FP), R0 ; how many digits to round after
CB AD E4 AD C2 0315 462 BSBB G_ROUND ; round the value
    FEAB 31 031A 463 SUBL2 DEC_EXP(FP), S_DF(FP) ; Alter format to fit
    031D 464 BRW F_CONVERT ; Finish with F conversion
    031D 465
    031D 466 ;+
    031D 467 ; G_ROUND rounds the value at the offset pointed to by R0.
    031D 468 ;-
    52 54 50 C1 031D 469 G_ROUND:
    35 62 91 0321 470 ADDL3 R0, R4, R2
    01 18 0324 471 CMPB (R2), #^A/5/ ; Round?
    05 0326 472 BGEQ 10$ ; Yes
    39 72 91 0327 473 RSB ; No, exit
    05 19 032A 474 10$: CMPB -(R2), #^A/9/ ; Is this a 9?
    62 30 90 032C 475 BLSS 20$ ; No, finish
    F6 11 032F 476 MOVB #^A/0/, (R2) ; Make it a zero
    62 96 0331 477 BRB 10$ ; Continue
    54 52 D1 0333 478 20$: INCB (R2) ; Round up
    06 18 0336 479 CMPL R2, R4 ; Have we moved past digit start?
    EC AD D7 0338 480 BGEQ 30$ ; No, exit
    E4 AD D6 033B 481 DECL STRING_ADDR(FP) ; Yes, move address
    05 033E 482 INCL DEC_EXP(FP) ; and exponent
    30$: RSB ; Exit

```

```

033F 485 .SBTTL INITIALIZE - Analyze argument list
033F 486 :+
033F 487 : INITIALIZE looks at the argument list and fills in the appropriate
033F 488 : values in the local frame.
033F 489 :-
033F 490
033F 491 INITIALIZE:
07 6C 91 033F 492 CMPB (AP), #7 ; 7 Arguments given?
20 13 0342 493 BEQL 70$ ; Skip defaults
F4 AD D4 0344 494 CLRL FLAGS(FP) ; No flags initially
CC AD 02 D0 0347 495 MOVL #2, S_DE(FP) ; Digits in exponent
D0 AD D4 0348 496 CLRL S_DI(FP) ; Digits in integer
D4 AD D4 034E 497 CLRL S_SCALE(FP) ; Scale factor
C8 AD D4 0351 498 CLRL S_DF(FP) ; Digits in fraction
05 02 6C 8F 0354 499 CASEB (AP), #2, #5 ; Select on number of arguments
0030' 0358 500 1$: .WORD 20$-1$ ; 2 arguments
002B' 035A 501 .WORD 30$-1$ ; 3 arguments
0026' 035C 502 .WORD 40$-1$ ; 4 arguments
0016' 035E 503 .WORD 50$-1$ ; 5 arguments
0011' 0360 504 .WORD 60$-1$ ; 6 arguments
000C' 0362 505 .WORD 70$-1$ ; 7 arguments
0364 506 : fall through ; assume >7 arguments
F4 AD 1C AC 9A 0364 507 70$: MOVZBL caller_flags(AP), FLAGS(FP)
CC AD 18 AC D0 0369 508 60$: MOVL digits_in_exp(AP), S_DE(FP)
D0 AD 14 AC D0 036E 509 50$: MOVL digits_in_int(AP), S_DI(FP)
D4 AD 10 AC D0 0373 510 MOVL scale_factor(AP), S_SCALE(FP)
C8 AD 0C AC D0 0378 511 MOVL digits_in_fract(AP), S_DF(FP) ; Digits in fraction part
05 037D 512 RSB
D0 AD 10 AC D0 037E 513 40$: MOVL scale_factor(AP), S_DI(FP) ; Release 1 meaning
C8 AD 0C AC D0 0383 514 30$: MOVL digits_in_fract(AP), S_DF(FP) ; Digits in fraction part
05 0388 515 20$: RSB

```



```

0389 517      .SBTTL INITIALIZE_F - Analyze argument list for F format
0389 518      ;+
0389 519      ; INITIALIZE_F is performs the same function as INITIALIZE except that
0389 520      ; the scale factor is truly the scale factor and that the digits_in_integer
0389 521      ; and digits_in_exp values are ignored.
0389 522      ;-
0389 523
0389 524 INITIALIZE F:
      F4 AD   D4 0389 525      CLRL   FLAGS(FP)           ; No flags initially
      D4 AC   D4 0389 526      CLRL   S_SCALE(FP)         ; Scale factor
C8 AD 04 03 6C 038F 527      MOVL   digits_in_fract(AP), S_DF(FP) ; Digits in fraction part
      04 03 6C 8F 0394 528      CASEB  (AP), #3, #4         ; Select on number of arguments
      0014' 0398 529 1$:      .WORD  30$-1$             ; 3 arguments
      000F' 039A 530          .WORD  40$-1$             ; 4 arguments
      000F' 039C 531          .WORD  50$-1$             ; 5 arguments
      000F' 039E 532          .WORD  60$-1$             ; 6 arguments
      000A' 03A0 533          .WORD  70$-1$             ; 7 arguments
      03A2 534          ; fall through                  ; assume >7 arguments
F4 AD 1C AC 90 03A2 535 70$:  MOVB   caller_flags(AP), FLAGS(FP)
      03A7 536 60$:
      03A7 537 50$:
D4 AD 10 AC D0 03A7 538 40$:  MOVL   scale_factor(AP), S_SCALE(FP)
      05 03AC 539 30$:      RSB
  
```

```

03AD 541 .SBTTL DIGITS_OUT
03AD 542 :+
03AD 543 : Routine to format the digits in the output string.
03AD 544 :
03AD 545 : The string will be constructed as follows:
03AD 546 :
03AD 547 : n1 blanks, where n1 is calculated
03AD 548 : LEAD_DIGITS digits
03AD 549 : a decimal point
03AD 550 : LEAD_ZERO zeroes
03AD 551 : TRAIL_DIGITS digits
03AD 552 :
03AD 553 : The sign is inserted where appropriate. If LEAD_DIGITS is
03AD 554 : zero, an optional leading zero is inserted if there is
03AD 555 : room.
03AD 556 :-
03AD 557 DIGITS_OUT:
56 EC AD D0 03AD 558 MOVL STRING_ADDR(FP), R6 ; Address of first digit
52 08 BC 7D 03B1 559 MOVQ @out_string(AP), R2 ; Get string descriptor
50 52 52 3C 03B5 560 MOVZWL R2, R2
50 52 C4 AD C3 03B8 561 SUBL3 LEAD_DIGITS(FP), R2, R0 ; Find leading blanks
50 50 C0 AD C2 03BD 562 SUBL2 LEAD_ZERO(FP), R0
50 50 BC AD C2 03C1 563 SUBL2 TRAIL_DIGITS(FP), R0
50 50 D8 AD 95 03C5 564 TSTB EXP_LETTER(FP) ; F format?
50 50 07 13 03C8 565 BEQL 10$ ; Yes
50 50 CC AD C2 03CA 566 SUBL2 S_DE(FP), R0
50 50 02 C2 03CE 567 SUBL2 #2, R0 ; Compensate for exponent
50 50 E8 AD D5 03D1 568 10$: TSTL SIGN(FP) ; Negative?
02 F4 AD 00 E1 03D4 569 BLSS 15$ ; Yes
50 50 D7 03DB 570 BBC #V_FORCEPLUS, FLAGS(FP), 20$ ; Force plus sign?
50 50 D7 03DD 571 15$: DECL R0 ; Use another character
50 50 66 19 03DF 572 20$: DECL R0 ; For decimal point
50 50 D0 03E1 573 BLSS ERROR ; No room left
50 50 06 13 03E4 574 MOVL R0, R7
50 50 20 90 03E6 575 BEQL 22$ ; Skip if no blanks
50 50 FA 50 F5 03E9 576 21$: MOVB #^A/ /, (R3)+ ; Insert leading blanks
50 50 E8 AD D5 03EC 577 SOBGR R0, 21$ ; Loop till done
08 F4 AD 00 E1 03EF 578 22$: TSTL SIGN(FP) ; Negative?
83 83 03 11 03F1 579 BLSS 25$ ; Yes
83 83 2B 90 03F6 580 BBC #V_FORCEPLUS, FLAGS(FP), 30$ ; Force + sign?
83 83 03 11 03F9 581 MOVB #^A/+/, (R3)+ ; Yes
83 83 2D 90 03FB 582 BRB 30$
83 83 C4 AD D5 03FE 583 25$: MOVB #^A/-/, (R3)+ ; Minus sign
83 83 22 14 0401 584 30$: TSTL LEAD_DIGITS(FP) ; Check for leading zero
83 83 BC AD D5 0403 585 BGTR 40$ ; Not necessary
83 83 0B 14 0406 586 TSTL TRAIL_DIGITS(FP) ; Required?
83 83 C0 AD 25 0408 587 BGTR 35$ ; No
83 83 06 14 040B 588 TSTL LEAD_ZERO(FP) ; Required?
83 83 57 D5 040D 589 BGTR 35$ ; No
83 83 36 15 040F 590 TSTL R7 ; Is there room?
83 83 04 11 0411 591 BLEQ ERROR ; No
83 83 57 D5 0413 592 BRB 37$ ; Put it in
83 83 0E 15 0415 593 35$: TSTL R7 ; Is there room?
83 83 20 73 91 0417 594 BLEQ 40$ ; No, skip it
83 83 FF A3 63 90 041A 595 37$: CMPB -(R3), #^A/ / ; Is last char a blank?
83 83 04 13 041A 596 BEQL 38$ ; Yes, dont move it
83 83 04 13 041C 597 MOVB (R3), -1(R3) ; Move sign

```

```

      83 30 90 0420 598 38$:  MOVB  #^A/0/, (R3)+      ; Insert zero
      08 11 0423 599
63 66 C4 AD 28 0425 600 40$:  MOVC3 LEAD_DIGITS(FP), (R6), (R3) ; Move leading digits
      56 51 D0 042A 601
      83 2E 90 042D 602 42$:  MOVB  #^A/./, (R3)+      ; Move decimal point
50 C0 AD D0 C430 603 42$:  MOVL  LEAD_ZERO(FP), R0      ; Insert leading zeroes
      06 15 0434 604
      83 30 90 0436 605 45$:  MOVB  #^A/0/, (R3)+      ; Move a zero
      FA 50 F5 0439 606
50 BC AD D0 043C 607 50$:  SOBGTR R0, 45$      ; Loop till done
      04 15 0440 608
63 66 50 28 0442 609 50$:  MOVL  TRAIL_DIGITS(FP), R0 ; Move trailing digits
      05 0446 610 60$:  BLEQ  60$      ; Skip if none
      0447 611
      0447 612 ERROR:
61 50 50 08 BC 7D 0447 613      MOVQ  @out_string(AP), R0
      2A 6E 00 2C 044B 614      MOVCS #0, (SP), #^A/*/, R0, (R1)
      50 00000000'8F D0 0451 615      MOVL  #FOR$_OUTCONERR, R0
      04 0458 616      RET

```

```
0459 618 .SBTTL FINISH
0459 619 ;+
0459 620 ; FINISH blank fills the remainder of the string and returns to the caller.
0459 621 ;-
0459 622 FINISH:
50 08 BC 7D 0459 623 MOVQ @out_string(AP), R0 ; Get string descriptor
50 50 3C 045D 624 MOVZWL R0, R0
51 50 C0 0460 625 ADDL2 R0, R1 ; Find last character
51 53 C2 0463 626 SUBL2 R3, R1 ; Subtract characters used
83 06 13 0466 627 BEQL EXIT ; If none left, exit
FA 20 90 0468 628 10$: MOVB #^A/ /, (R3)+ ; Move a blank
50 FA 51 F5 046B 629 SOBGTR R1, 10$ ; Loop till done
04 04 00 046E 630 EXIT: MOVL #1, R0 ; $$$_NORMAL
0471 631 RET
0472 632
0472 633
0472 634 .END
```

FOR\$CVTRT
Symbol table

- Convert Real (F, D, G, H) to Text F 12

15-SEP-1984 23:49:35
6-SEP-1984 10:54:25

VAX/VMS Macro V04-00
[FORRTL.SRC]FOR\$CVTRT.MAR;1

Page 18
(10)

```

CALLER_FLAGS = 0000001C
COMMON_E     = 000000A1 R 01
COMMON_F     = 0000018A R 01
COMMON_FRAME = FFFFFFFDC
COMMON_G     = 00000256 R 01
DEC_EXP      = FFFFFFFE4
DIGITS_IN_EXP = 00000018
DIGITS_IN_FRACT = 0000000C
DIGITS_IN_INT = 00000014
DIGITS_OUT   = 000003AD R 01
ERROR        = 00000447 R 01
EXIT         = 0000046E R 01
EXP_LETTER   = FFFFFFFD8
E_CONVERT    = 000000DC R 01
FG_ERROR     = 00000203 R 01
FINISH       = 00000459 R 01
FLAGS        = FFFFFFFF4
FOR$CNV_OUT_D = 00000065 RG 01
FOR$CNV_OUT_E = 00000015 RG 01
FOR$CNV_OUT_F = 0000015D RG 01
FOR$CNV_OUT_G = 0000021A RG 01
FOR$CVT_D_TD  = 00000065 RG 01
FOR$CVT_D_TE  = 00000015 RG 01
FOR$CVT_D_TF  = 0000015D RG 01
FOR$CVT_D_TG  = 0000021A RG 01
FOR$CVT_F_TD  = 00000051 RG 01
FOR$CVT_F_TE  = 00000000 RG 01
FOR$CVT_F_TF  = 0000014E RG 01
FOR$CVT_F_TG  = 00000206 RG 01
FOR$CVT_G_TD  = 00000079 RG 01
FOR$CVT_G_TE  = 00000029 RG 01
FOR$CVT_G_TF  = 0000016C RG 01
FOR$CVT_G_TG  = 0000022E RG 01
FOR$CVT_H_TD  = 0000008D RG 01
FOR$CVT_H_TE  = 0000003D RG 01
FOR$CVT_H_TF  = 0000017B RG 01
FOR$CVT_H_TG  = 00000242 RG 01
FOR$OUTCONERR = ***** X 00
FRAME        = FFFFFFFBC
F_CONVERT    = 000001C8 R 01
G_ROUND      = 0000031D R 01
INITIALIZE   = 0000033F R 01
INITIALIZE_F = 00000389 R 01
LEAD_DIGITS  = FFFFFFFC4
LEAD_ZERO    = FFFFFFFC0
M_RT_ROUND   = 02000000
M_TRUNCATE   = 01000000
OFFSET       = FFFFFFFE0
OTSS$CVT_D_T_R8 = ***** X 00
OTSS$CVT_F_T_R8 = ***** X 00
OTSS$CVT_G_T_R8 = ***** X 00
OTSS$CVT_H_T_R8 = ***** X 00
OUT_STRING   = 00000008
PACKED       = FFFFFFFF8
REGMASK      = 000001FC
RT_RND       = FFFFFFFDC
SCALE_FACTOR = 00000010
    
```

```

SIGN          = FFFFFFFE8
SIG_DIGITS    = FFFFFFFF0
STRING_ADDR   = FFFFFFFEC
S_DE          = FFFFFFFCC
S_DF          = FFFFFFFC8
S_DI          = FFFFFFFD0
S_SCALE       = FFFFFFFD4
TRAIL_DIGITS  = FFFFFFFBC
TRY_F         = 000002CC R R 01
USE_E         = 000002F7 R R 01
USE_F         = 0000030F R R 01
VALUE         = 00000004
V_ERR_OFLO    = 00000001
V_FORCEPLUS   = 00000000
    
```

! Psect synopsis !

PSECT name	Allocation	PSECT No.	Attributes
ABS	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
_FOR\$CODE	00000472 (1138.)	01 (1.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

! Performance indicators !

Phase	Page faults	CPU Time	Elapsed Time
Initialization	33	00:00:00.10	00:00:00.52
Command processing	117	00:00:00.47	00:00:03.44
Pass 1	99	00:00:02.18	00:00:05.71
Symbol table sort	0	00:00:00.08	00:00:00.16
Pass 2	125	00:00:01.39	00:00:04.23
Symbol table output	7	00:00:00.06	00:00:00.06
Psect synopsis output	3	00:00:00.03	00:00:00.04
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	386	00:00:04.31	00:00:14.16

The working set limit was 1050 pages.
13377 bytes (27 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 71 non-local and 46 local symbols.
634 source lines were read in Pass 1, producing 57 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\$:FOR\$CVTRT/OBJ=OBJ\$:FOR\$CVTRT MSRCS\$:FOR\$CVTRT/UPDATE=(ENHS\$:FOR\$CVTRT)

Grid of software modules including: COMR50WD LIS, FORDETS LIS, FORDECOM LIS, COMSET LIS, FORASSOC LIS, FORB LIS, FORCLOSE LIS, FORDATE LIS, FORCLOSE LIS, FORDECOMP LIS, FORDELETE LIS, COMRAD50 LIS, COMUSEREX LIS, FORBITOPS LIS, FORBACKSP LIS, FORCUT LIS, and FORDISP LIS.