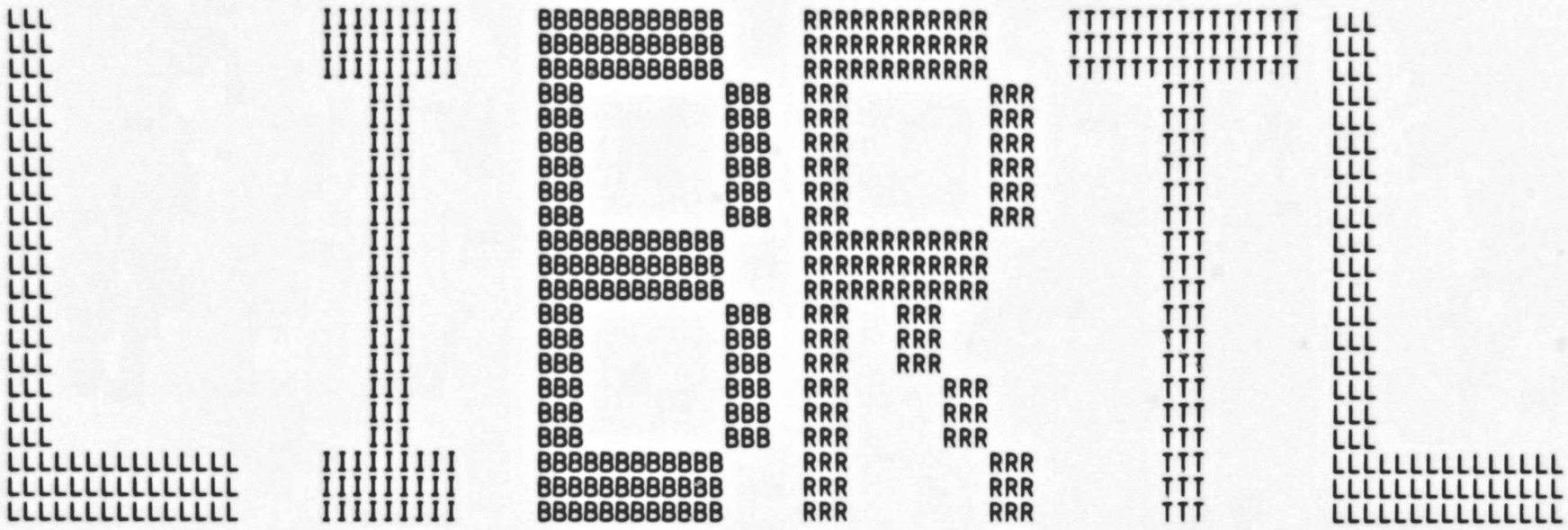
LLL IIIIIIIII BBBBBBBBBBB RRRRRRRRRRR TTTTTTTTTTTTTTT LLL
LLL IIIIIIIII BBBBBBBBBBBB RRRRRRRRRRRR TTTTTTTTTTTTTTT LLL
LLL IIIIIIIII BBBBBBBBBBBB RRRRRRRRRRR TTTTTTTTTTTTTTT LLL
LLL III BBB BBB RRR RRR TTT LLL
LLL III BBB BBB RRR RRR TTT LLL
LLL III BBB BBB RRR RRR TTT LLL
LLL III BBB BBB RRR RRR TTT LLL
LLL III BBB BBB RRR RRR TTT LLL
LLL III BBB BBB RRR RRR TTT LLL
LLL III BBBBBBBBBBB RRRRRRRRRRR 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
LLLLLLLLLLLLLLL IIIIIIIII BBBBBBBBBBBB RRR RRR TTT LLLLLLLLLLLLLLL
LLLLLLLLLLLLLLL IIIIIIIII BBBBBBBBBBBB RRR RRR TTT LLLLLLLLLLLLLLL
LLLLLLLLLLLLLLL IIIIIIIII BBBBBBBBBBBB RRR RRR TTT LLLLLLLLLLLLLLL

```
  OOOOOO    TTTTTTTTTT    SSSSSSSS    CCCCCCCC  VV      VV  TTTTTTTTTT  RRRRRRRR      GGGGGGGG  PPPPPPPP
  OOOOOO    TTTTTTTTTT    SSSSSSSS    CCCCCCCC  VV      VV  TTTTTTTTTT  RRRRRRRR      GGGGGGGG  PPPPPPPP
OO      OO      TT      SS          CC          VV      VV      TT      RR      RR  GG            PP      PP
OO      OO      TT      SS          CC          VV      VV      TT      RR      RR  GG            PP      PP
OO      OO      TT      SS          CC          VV      VV      TT      RR      RR  GG            PP      PP
OO      OO      TT      SS          CC          VV      VV      TT      RR      RR  GG            PP      PP
OO      OO      TT        SSSSSS    CC          VV      VV      TT      RRRRRRRR    GG            PPPPPPPP
OO      OO      TT        SSSSSS    CC          VV      VV      TT      RRRRRRRR    GG            PPPPPPPP
OO      OO      TT              SS  CC          VV      VV      TT      RR   RR     GG   GGGGGG   PP
OO      OO      TT              SS  CC          VV      VV      TT      RR   RR     GG   GGGGGG   PP
OO      OO      TT              SS  CC            VV  VV        TT      RR    RR    GG      GG    PP
OO      OO      TT              SS  CC            VV  VV        TT      RR    RR    GG      GG    PP
  OOOOOO        TT      SSSSSSSS      CCCCCCCC      VV          TT      RR      RR    GGGGGG     PP
  OOOOOO        TT      SSSSSSSS      CCCCCCCC      VV          TT      RR      RR    GGGGGG     PP

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
```

Table of contents

```
0000     1          .TITLE  OTS$CVTRGP_R9   Convert Rounded G Floating to Packed
0000     2          .IDENT  /1-005/         ; File: OTSCVTRGP.MAR Edit: DG1005
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  ; FACILITY: LANGUAGE INDEPENDENT SUPPORT
0000    29  ;++
0000    30  ; ABSTRACT:
0000    31  ;       This module contains the routine that converts g floating
0000    32  ;       numbers to packed with rounding.
0000    33  ;
0000    34  ;
0000    35  ;--
0000    36  ;
0000    37  ; VERSION: 1
0000    38  ;
0000    39  ; HISTORY:
0000    40  ;
0000    41  ; AUTHOR:
0000    42  ;       Pamela Levesque, 21-Jan-1982
0000    43  ;
0000    44  ; MODIFIED BY:
0000    45  ;
0000    46  ;
0000    47  ;
```

```
0000     49          .SBTTL  HISTORY         ; Detailed Current Edit History
0000     50
0000     51
0000     52 ; Edit History for Version 1 of OTSCVTRGP
0000     53 ;
0000     54 ; 1-001 - Original.  PLL 21-Jan-1982
0000     55 ; 1-002 - Call g floating conversion routine.  PLL 2-Mar-1982
0000     56 ; 1-003 - Expect a 2 digit exponent from the conversion routine.  PLL 8-Mar-1982
0000     57 ; 1-004 - determine how many digits there are in the exponent on the fly.
0000     58 ;         MDL 9-Apr-1984
0000     59 ; 1-005 - Make sure result will not overflow destination after calculating
0000     60 ;         longword exponent.  DG 9-Apr-1984
```

```
                        .SBTTL  DECLARATIONS

;
; INCLUDE FILES:
;
                        $DSCDEF

;
; EXTERNAL SYMBOLS:

                        .DSABL  GBL             ; Prevent undeclared symbols from being
                                                ; automatically global

                        .EXTRN  OTS$CNVOUT_G    ; E-format conversion
                        .EXTRN  LIB$STOP        ; Error halt
                        .EXTRN  OTS$_FATINTERR  ; Fatal internal error code
;

;
; MACROS:
;       NONE
;

;
; PSECT DECLARATIONS:
00000000                .PSECT  _OTS$CODE       PIC, SHR, LONG, EXE, NOWRT,-
                                                USR, CON, REL, LCL, RD

;
; EQUATED SYMBOLS:
;       NONE
;

;
; OWN STORAGE:
;       NONE
;
```

```
                                                D 5
0000    100          .SBTTL  OTS$CVTRGP_R9
0000    101
0000    102 ;++
0000    103 ; FUNCTIONAL DESCRIPTION:
0000    104 ;
0000    105 ;       Converts a g floating number to packed with rounding.
0000    106 ;
0000    107 ; CALLING SEQUENCE:
0000    108 ;
0000    109 ;       JSB OTS$CVTRGP_R9 (scale.rl.v, src.rg.r, dstlen.rl.v, dst.wp.r)
0000    110 ;
0000    111 ;       Arguments are passed in R6, R7, R8 and R9.
0000    112 ;
0000    113 ; INPUT PARAMETERS:
0000    114 ;
0000    115 ;       SCALE.rl.v              The power of ten by which the internal
0000    116 ;                               representation of the source must be
0000    117 ;                               multiplied to scale the same as the
0000    118 ;                               internal representation of the dest.
0000    119 ;       SRC.rg.r                The number to be converted
0000    120 ;       DSTLEN.rl.v             The number of digits in the destination
0000    121 ;
0000    122 ; IMPLICIT INPUTS:
0000    123 ;
0000    124 ;       All of the trap bits in the PSL are assumed off.
0000    125 ;
0000    126 ; OUTPUT PARAMETERS:
0000    127 ;
0000    128 ;       DST.wp.r                The place to store the converted number
0000    129 ;
0000    130 ; IMPLICIT OUTPUTS:
0000    131 ;
0000    132 ;       NONE
0000    133 ;
0000    134 ; FUNCTION VALUE:
0000    135 ;
0000    136 ;       1 = SUCCESS, 0 = FAILURE
0000    137 ;
0000    138 ; SIDE EFFECTS:
0000    139 ;
0000    140 ;       Destroys registers R0 through R9.
0000    141 ;
0000    142 ;--
0000    143
0000    144
0000    145 OTS$CVTRGP_R9::
      5E   30   C2 0000    146          SUBL2   #48,SP                          ; Allocate temp space
6E    30   00   6E 00   2C 0003    147          MOVC5   #0,(SP),#^A'',#48,(SP)  ; Clear temp space
      6E   67 50FD 0009    148          MOVG    (R7),(SP)                       ; Get input number
                   000D    149 ;
                   000D    150 ; Make a descriptor for the temporary string.
                   000D    151 ;
        08 AE   9F 000D    152          PUSHAB  8(SP)                           ; Address = space reserved
      7E   01   90 0010    153          MOVB    #DSC$K_CLASS_S,-(SP)            ; Class = static
      7E   0E   90 0013    154          MOVB    #DSC$K_DTYPE_T,-(SP)            ; Data type = ASCII string
      7E   26   B0 0016    155          MOVW    #38,-(SP)                       ; Length in bytes
                   0019    156 ;
```

```
                                0019   157 ; Call OTS$CNVOUT.
                                0019   158 ;
                          1F    DD 0019   159         PUSHL   #31                     ; Digits in fraction
                       04 AE    9F 001B   160         PUSHAB  4(SP)                   ; Output string descriptor
                       10 AE    9F 001E   161         PUSHAB  16(SP)                  ; Number to convert
          00000000'GF    03    FB 0021   162         CALLS   #3,G^OTS$CNVOUT_G       ; Call conversion routine
                    5B   50    E9 0028   163         BLBC    R0,20$                  ; Should never fail
                                002B   164 ;
                                002B   165 ; Determine the number of digits in the exponent.
                                002B   166 ;
                                002B   167                                         ; find beginning of exponent (sign byte)
                 14 AE    24   2B 3A 002B   168         LOCC    #^A'+',#36,20(SP)       ; look for sign byte (+)
                          07   12 0030   169         BNEQ    5$                      ; branch if we found it
                 14 AE    24   2D 3A 0032   170         LOCC    #^A'-',#36,20(SP)       ; look for sign byte (-)
                          4D   13 0037   171         BEQL    20$                     ; branch to error if we didn't find it
                                0039   172                                         ;
                    08 AE    51   D0 0039   173 5$:     MOVL    R1,8(SP)                ; save location of sign byte
                                003D   174                                         ; (contained in R1 after above LOCC)
                                003D   175                                         ;
              08 BE    05    00   3A 003D   176         LOCC    #^A'',#5,@B^8(SP)       ; look for NUL (end of string)
           0C AE    51    08 AE    C3 0042   177         SUBL3   8(SP),R1,12(SP)         ; determine number of digits by subtracting
                                0048   178                                         ; location of beginning of exponent from
                                0048   179                                         ; location of end of exponent, and save it
                    0C AE    D7   D7 0048   180         DECL    12(SP)                  ; adjust for sign byte
                                004B   181                                         ; at this point, 12(SP) = length of exponent
                                004B   182 ;
                                004B   183 ; Convert the exponent and correct for scale factor.
                                004B   184 ;
   6E    0C AE    08 BE    0C AE    09 004B   185         CVTSP   12(SP),@B^8(SP),12(SP),(SP)     ; Make packed exponent
              50    6E    0C AE    36 0053   186         CVTPL   12(SP),(SP),R0          ; Make longword exponent
                                0058   187                 ;
                                0058   188                 ; Here must check that the result will not overflow
                                0058   189                 ; the destination.  The destination length (total number of
                                0058   190                 ; digits) minus the scale factor will give the number of digits
                                0058   191                 ; to the left of the decimal point.  If that number is greater than
                                0058   192                 ; the longword exponent, then the result WILL overflow the dest.
                                0058   193                 ;
                    54    58    56   C3 0058   194         SUBL3   R6,R8,R4                ; Determine num of digits on left
                          54    50   D1 005C   195         CMPL    R0,R4                   ; Compare with longword exponent
                                1F   14 005F   196         BGTR    15$                     ; Branch if overflowed
                                0061   197
                    56    E1 A640    9E 0061   198         MOVAB   -31(R6)[R0],R6          ; Correct for fraction size and scale
                                0066   199 ;
                                0066   200 ; Convert the fraction to packed.
                                0066   201 ;
                 12 AE    10 AE    90 0066   202         MOVB    16(SP),18(SP)           ; Move sign over "."
        6E    1F    12 AE    1F    09 006B   203         CVTSP   #31,18(SP),#31,(SP)     ; Make packed fraction
69   58   05    6E    1F    56    F8 0071   204         ASHP    R6,#31,(SP),#5,R8,(R9)  ; Scale to destination
                                0078   205                 ;       (also clears R0)
                                02   1D 0078   206         BVS     10$                     ; Branch if overflowed
                                50   D6 007A   207         INCL    R0                      ; Indicate success, R0 = 1
                          5E    38   C0 007C   208 10$:     ADDL2   #56,SP                  ; Delete stack temps
                                05   05 007F   209         RSB
                                0080   210
                                50   D4 0080   211 15$:     CLRL    R0                      ; Indicate failure, R0 = 0
                          5E    38   C0 0082   212         ADDL2   #56,SP                  ; Delete stack temps
                                05   05 0085   213         RSB
```

```
                     0086   214 ;
                     0086   215 ; Come here on failure of OTS$CNVOUT.  This should never happen.
                     0086   216 ;
     00000000'8F   DD  0086   217 20$:    PUSHL   #OTS$_FATINTERR          ; OTS fatal error message
00000000'GF    01   FB  008C   218         CALLS   #1,G^LIB$STOP            ; Signal and don't return
                     0093   219 ;
                     0093   220         .END
```

```
DSC$K_CLASS_S                   = 00000001
DSC$K_DTYPE_T                   = 0000000E
LIB$STOP                        ********    X   00
OTS$CNVOUT_G                    ********    X   00
OTS$CVTRGP_R9                   00000000 RG     02
OTS$_FATINTERR                  ********    X   00
```

### 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 |
| _OTS$CODE | 00000093 ( 147.) | 02 ( 2.) | PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG |

### Performance indicators

| Phase | Page faults | CPU Time | Elapsed Time |
|---|---|---|---|
| Initialization | 35 | 00:00:00.05 | 00:00:01.25 |
| Command processing | 129 | 00:00:00.29 | 00:00:01.49 |
| Pass 1 | 132 | 00:00:01.19 | 00:00:06.66 |
| Symbol table sort | 0 | 00:00:00.09 | 00:00:00.94 |
| Pass 2 | 57 | 00:00:00.38 | 00:00:01.44 |
| Symbol table output | 2 | 00:00:00.01 | 00:00:00.01 |
| 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 | 359 | 00:00:02.02 | 00:00:11.80 |

The working set limit was 1050 pages.
9075 bytes (18 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 135 non-local and 4 local symbols.
220 source lines were read in Pass 1, producing 10 object records in Pass 2.
8 pages of virtual memory were used to define 7 macros.

### Macro library statistics

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

190 GETS were required to define 4 macros.

There were no errors, warnings or information messages.

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

OTSCVTPD
LIS

OTSCVTTF
LIS

OTSCVTRFP
LIS

OTSCVTRT
LIS

OTSCVTTOL
LIS

OTSCVTPG
LIS

OTSCVTTR
LIS

OTSCVTRHP
LIS

OTSLINKAG
LIS

OTSCVTRDP
LIS

OTSCVTTIL
LIS

OTSCVTTLL
LIS

OTSCVTPF
LIS

OTSCVTRGP
LIS

OTSMOVE
LIS

OTSMSG
LIS

OTSCVTPH
LIS

OTSLUN
LIS