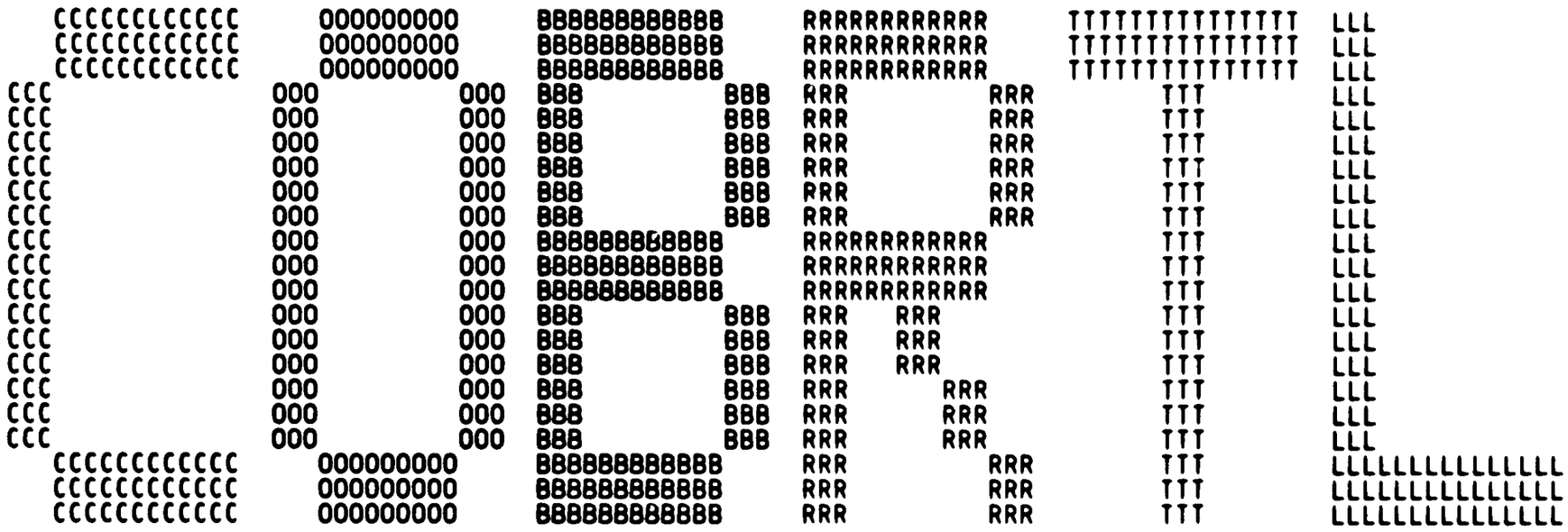
CCCCCCCCCCC 000000000 BBBBBBBBBBBB RRRRRRRRRRR TTTTTTTTTTTTTTT LLL
CCCCCCCCCCC 000000000 BBBBBBBBBBBB RRRRRRRRRRR TTTTTTTTTTTTTTT LLL
CCCCCCCCCCC 000000000 BBBBBBBBBBBB RRRRRRRRRRR TTTTTTTTTTTTTTT LLL
CCC 000 000 BBB BBB RRR RRR TTT LLL
CCC 000 000 BBB BBB RRR RRR TTT LLL
CCC 000 000 BBB BBB RRR RRR TTT LLL
CCC 000 000 BBB BBB RRR RRR TTT LLL
CCC 000 000 BBB BBB RRR RRR TTT LLL
CCC 000 000 BBB BBB RRR RRR TTT LLL
CCC 000 000 BBBBBBBBBBB RRRRRRRRRRR TTT LLL
CCC 000 000 BBBBBBBBBBBB RRRRRRRRRRR TTT LLL
CCC 000 000 BBBBBBBBBBBB RRRRRRRRRRR TTT LLL
CCC 000 000 BBB BBB RRR RRR TTT LLL
CCC 000 000 BBB BBB RRR RRR TTT LLL
CCC 000 000 BBB BBB RRR RRR TTT LLL
CCC 000 000 BBB BBB RRR RRR TTT LLL
CCC 000 000 BBB BBB RRR RRR TTT LLL
CCC 000 000 BBB BBB RRR RRR TTT LLL
CCCCCCCCCCC 000000000 BBBBBBBBBBBB RRR RRR TTT LLLLLLLLLLLLLLL
CCCCCCCCCCC 000000000 BBBBBBBBBBBB RRR RRR TTT LLLLLLLLLLLLLLL
CCCCCCCCCCC 000000000 BBBBBBBBBBBB RRR RRR TTT LLLLLLLLLLLLLLL

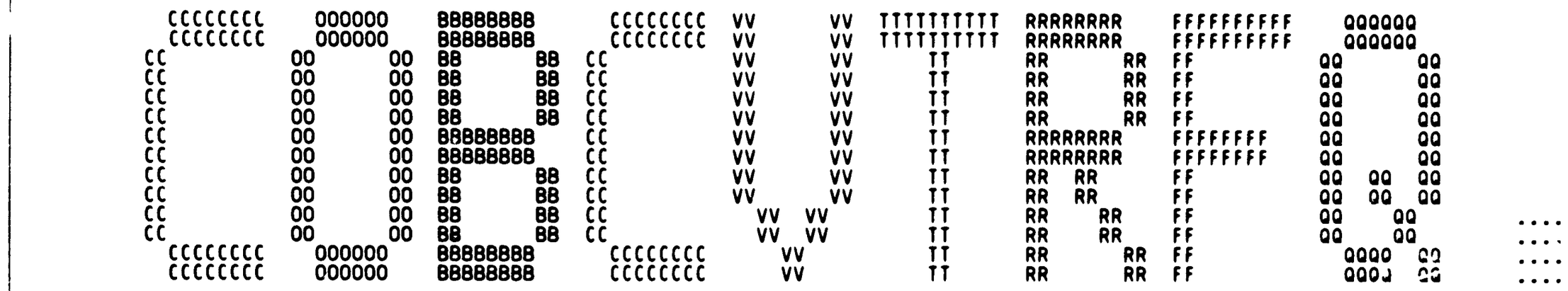
CCCCCCCC 000000 BBBBBBBB CCCCCCCC VV VV TTTTTTTTTT RRRRRRRR FFFFFFFFFF QQQQQQ
CCCCCCCC 000000 BBBBBBBB CCCCCCCC VV VV TTTTTTTTTT RRRRRRRR FFFFFFFFFF QQQQQQ
CC CC OO OO BB BB CC CC VV VV TT RR RR FF QQ QQ
CC CC OO OO BB BB CC CC VV VV TT RR RR FF QQ QQ
CC CC OO OO BB BB CC CC VV VV TT RR RR FF QQ QQ
CC CC OO OO BB BB CC CC VV VV TT RR RR FF QQ QQ
CC CC OO OO BBBBBBBB CC CC VV VV TT RRRRRRRR FFFFFFFF QQ QQ
CC CC OO OO BBBBBBBB CC CC VV VV TT RRRRRRRR FFFFFFFF QQ QQ
CC CC OO OO BB BB CC CC VV VV TT RR RR FF QQ QQ QQ
CC CC OO OO BB BB CC CC VV VV TT RR RR FF QQ QQ QQ
CC CC OO OO BB BB CC CC VV VV TT RR RR FF QQ QQ
CC CC OO OO BB BB CC CC VV VV TT RR RR FF QQ QQ
CCCCCCCC 000000 BBBBBBBB CCCCCCCC VV TT RR RR FF QQQQ QQ
CCCCCCCC 000000 BBBBBBBB CCCCCCCC VV TT RR RR FF QQQQ QQ
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

```
                        .TITLE  COB$CVTRFQ_R8   COBOL Convert Rounded Floating to Quadword
                        .IDENT  /1-006/         ; File: COBCVTRFQ.MAR

                ;
                ;*****************************************************************************
                ;*                                                                           *
                ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
                ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
                ;*  ALL RIGHTS RESERVED.                                                     *
                ;*                                                                           *
                ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
                ;*  ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE           *
                ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER      *
                ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
                ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY       *
                ;*  TRANSFERRED.                                                             *
                ;*                                                                           *
                ;*  THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE     *
                ;*  AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT         *
                ;*  CORPORATION.                                                             *
                ;*                                                                           *
                ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS      *
                ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
                ;*                                                                           *
                ;*                                                                           *
                ;*****************************************************************************
                ;
                ; FACILITY: COBOL SUPPORT
                ;++
                ; ABSTRACT:
                ;       This module contains the routine that converts floating
                ;       numbers to quadword with rounding.
                ;
                ;
                ;--
                ;
                ; VERSION: 1
                ;
                ; HISTORY:
                ;
                ; AUTHOR:
                ;       Marty Jack, 14-Mar-1979
                ;
                ; MODIFIED BY:
                ;
                ;
```

```
0000     49          .SBTTL  HISTORY         ; Detailed Current Edit History
0000     50
0000     51
0000     52 ; Edit History for Version 1 of COBCVTRFQ
0000     53 ;
0000     54 ; 1-001 - Original.  MLJ 14-Mar-1979
0000     55 ; 1-002 - Correct problem with high order longword.  MLJ 22-Mar-1979
0000     56 ; 1-003 - Correct round toward zero problem.  PDG 1    n-1979
0000     57 ; 1-004 - Make external references explicit. RKR 17-      079
0000     58 ; 1-005 - Change all references to FOR$CNV_OUT_E to COB$CNVOUT
0000     59 ;         RKR  27-SEPT-79
0000     60 ; 1-006 - Cosmetic changes. RKR 18-OCT-79
```

```
                     0000     62          .SBTTL  DECLARATIONS
                     0000     63
                     0000     64 ;
                     0000     65 ; INCLUDE FILES:
                     0000     66 ;
                     0000     67          $DSCDEF
                     0000     68
                     0000     69 ;
                     0000     70 ; EXTERNAL SYMBOLS:
                     0000     71
                     0000     72          .DSABL  GBL             ; Prevent undeclared symbols from being
                     0000     73                                  ; automatically global.
                     0000     74
                     0000     75          .EXTRN  COB$CNVOUT      ; E-format conversion
                     0000     76          .EXTRN  LIB$STOP        ; Error halt
                     0000     77          .EXTRN  OTS$_FATINTERR  ; Fatal internal error code
                     0000     78 ;
                     0000     79
                     0000     80 ;
                     0000     81 ; MACROS:
                     0000     82 ;        NONE
                     0000     83 ;
                     0000     84
                     0000     85 ;
                     0000     86 ; PSECT DECLARATIONS:
                00000000     87          .PSECT  _COB$CODE       PIC, SHR, LONG, EXE, NOWRT
                     0000     88
                     0000     89 ;
                     0000     90 ; EQUATED SYMBOLS:
                     0000     91 ;        NONE
                     0000     92 ;
                     0000     93
                     0000     94 ;
                     0000     95 ; OWN STORAGE:
                     0000     96 ;
                     0000     97 ;+
                     0000     98 ; The following constant has the value 2**32.  It is used for scaling
                     0000     99 ; the high 32 bits and for compensating for unsigned arithmetic.
                     0000    100 ;-
6C 29 67 49 29 04    0000    101 BIAS:   .PACKED 4294967296
                     0006    102 ;+
                     0006    103 ; The following constant is 2**32-1.  It is subtracted from negative numbers,
                     0006    104 ; to compensate for DIVP truncating towards zero.
                     0006    105 ;-
5C 29 67 49 29 04    0006    106 BIAS_1: .PACKED 4294967295
           0000000A  000C    107 BIAS_DIGITS=10
                     000C    108 ;
```

```
                     000C   110          .SBTTL  COB$CVTRFQ_R8
                     000C   111
                     000C   112 ;++
                     000C   113 ; FUNCTIONAL DESCRIPTION:
                     000C   114 ;
                     000C   115 ;       Converts a floating number to quadword with rounding.
                     000C   116 ;
                     000C   117 ; CALLING SEQUENCE:
                     000C   118 ;
                     000C   119 ;       JSB COB$CVTRFQ_R8 (scale.rl.v, src.rf.r, dst.wq.r)
                     000C   120 ;
                     000C   121 ;       Arguments are passed in R6, R7, and R8.
                     000C   122 ;
                     000C   123 ; INPUT PARAMETERS:
                     000C   124 ;
                     000C   125 ;       SCALE.rl.v              The power of ten by which the internal
                     000C   126 ;                               representation of the source must be
                     000C   127 ;                               multiplied to scale the same as the
                     000C   128 ;                               internal representation of the dest.
                     000C   129 ;       SRC.rf.r                The number to be converted
                     000C   130 ;
                     000C   131 ; IMPLICIT INPUTS:
                     000C   132 ;
                     000C   133 ;       All of the trap bits in the PSL are assumed off.
                     000C   134 ;
                     000C   135 ; OUTPUT PARAMETERS:
                     000C   136 ;
                     000C   137 ;       DST.wq.r                The place to store the converted number
                     000C   138 ;
                     000C   139 ; IMPLICIT OUTPUTS:
                     000C   140 ;
                     000C   141 ;       NONE
                     000C   142 ;
                     000C   143 ; FUNCTION VALUE:
                     000C   144 ;
                     000C   145 ;       1 = SUCCESS, 0 = FAILURE
                     000C   146 ;
                     000C   147 ; SIDE EFFECTS:
                     000C   148 ;
                     000C   149 ;       Destroys registers R0 through R8.
                     000C   150 ;
                     000C   151 ;--
                     000C   152
                     000C   153
                     000C   154 COB$CVTRFQ_R8::
5E   30   C2         000C   155          SUBL2   #48,SP                  ; Allocate temp space
6E   67   56         000F   156          CVTFD   (R7),(SP)               ; Get input number
                     0012   157 ;
                     0012   158 ; Make a descriptor for the temporary string.
                     0012   159 ;
  08 AE   9F         0012   160          PUSHAB  8(SP)                   ; Address = space reserved
7E   01   90         0015   161          MOVB    #DSC$K_CLASS_S,-(SP)    ; Class = static
7E   0E   90         0018   162          MOVB    #DSC$K_DTYPE_T,-(SP)    ; Data type = ASCII string
7E   26   B0         001B   163          MOVW    #38,-(SP)               ; Length in bytes
                     001E   164 ;
                     001E   165 ; Call COB$CNVOUT.
                     001E   166 ;
```

```
                                 1F   DD  001E  167          PUSHL   #31                             ; Digits in fraction
                              04 AE   9F  0020  168          PUSHAB  4(SP)                           ; Output string descriptor
                              10 AE   9F  0023  169          PUSHAB  16(SP)                          ; Number to convert
                   00000000'GF    03   FB  0026  170          CALLS   #3,G^COB$CNVOUT                 ; Call conversion routine
                              55 50   E9  002D  171          BLBC    R0,20$                          ; Should never fail
                                      0030  172 ;
                                      0030  173 ; Convert the exponent and correct for scale factor.
                                      0030  174 ;
          6E    02    33 AE    02    09  0030  175          CVTSP   #2,51(SP),#2,(SP)               ; Make packed exponent
                   50    6E    02    36  0036  176          CVTPL   #2,(SP),R0                      ; Make longword exponent
                   56    E1 A640     9E  003A  177          MOVAB   -31(R6)[R0],R6                  ; Correct for fraction size and scale
                                      003F  178 ;
                                      003F  179 ; Convert the fraction to packed.
                                      003F  180 ;
                12 AE    10 AE    90  003F  181          MOVB    16(SP),18(SP)                   ; Move sign over "."
          6E    1F    12 AE    1F    09  0044  182          CVTSP   #31,18(SP),#31,(SP)             ; Make packed fraction
10 AE    13    05    6E    1F    56    F8  004A  183          ASHP    R6,#31,(SP),#5,#19,16(SP) ; Scale to temporary
                                      0052  184                                                  ;     (also clears R0)
                                 0D   1D  0052  185          BVS     10$                             ; Branch if won't fit in 19 digits
                   68    63    13    36  0054  186          CVTPL   #19,(R3),(R8)                   ; Convert to longword
                                      0058  187                                                  ;     (also clears R0)
                                 0B   1D  0058  188          BVS     11$                             ; Can't fit in 32 bits
             68    88    E1    8F    78  005A  189          ASHL    #-31,(R8)+,(R8)                 ; Spread sign bit
                                 50   D6  005F  190          INCL    R0                              ; Indicate success, R0 = 1
                         5E    38    C0  0061  191 10$:      ADDL2   #56,SP                          ; Delete stack temps
                                 05   0064  192          RSB
                                      0065  193 ;
                                      0065  194 ; Come here if the packed number won't fit in 32 bits.
                                      0065  195 ; Divide by 2**32 to get the high 32 bits of the quadword.
                                      0065  196 ;
                         06 09 A1    E9  0065  197 11$:      BLBC    9(R1),13$                       ; Skip if positive
       61    13    99 AF    0A    22  0069  198          SUBP4   #BIAS_DIGITS,BIAS_1,#19,(R1)    ; Make more negative
13    10 AE    13    8D AF    0A    27  006F  199 13$:      DIVP    #BIAS_DIGITS,BIAS,#19,16(SP),#19,(SP)  ; Divide by 2**32
                                 6E   0077
             04 A8    6E    13    36  0078  200          CVTPL   #19,(SP),4(R8)                   ; Convert and store high bits
                                      007D  201                                                  ;     (also clears R0)
                                 02   1D  007D  202          BVS     12$                             ; Number too large for a 64-bit integer
                                 50   D6  007F  203          INCL    R0                              ; Indicate success, R0 = 1
                         5E    38    C0  0081  204 12$:      ADDL2   #56,SP                          ; Delete stack temps
                                 05   0084  205          RSB                                     ; Return
                                      0085  206 ;
                                      0085  207 ; Come here on failure of COB$CNVOUT.  This should never happen.
                                      0085  208 ;
                00000000'8F    DD    0085  209 20$:      PUSHL   #OTS$_FATINTERR                 ; OTS fatal error message
             00000000'GF    01    FB  008B  210          CALLS   #1,G^LIB$STOP                   ; Signal and don't return
                                      0092  211 ;
                                      0092  212          .END
```

```
BIAS                            00000000 R      02
BIAS_1                          00000006 R      02
BIAS_DIGITS                   = 0000000A
COB$CNVOUT                      ********   X    00
COB$CVTRFQ_R8                   0000000C RG     02
DSC$K_CLASS_S                 = 00000001
DSC$K_DTYPE_T                 = 0000000E
LIB$STOP                        ********   X    00
OTS$_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
_COB$CODE	00000092 (146.)	02 (2.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	30	00:00:00.04	00:00:01.91
Command processing	117	00:00:00.35	00:00:02.61
Pass 1	137	00:00:01.21	00:00:08.24
Symbol table sort	0	00:00:00.11	00:00:00.11
Pass 2	51	00:00:00.39	00:00:02.17
Symbol table output	2	00:00:00.02	00:00:00.02
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	342	00:00:02.13	00:00:15.13

```
The working set limit was 900 pages.
9011 bytes (18 pages) of virtual memory were used to buffer the intermediate code.
There were 10 pages of symbol table space allocated to hold 138 non-local and 5 local symbols.
212 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
_255DUA28:[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$:COBCVTRFQ/OBJ=OBJ$:COBCVTRFQ MSRC$:COBCVTRFQ/UPDATE=(ENH$:COBCVTRFQ)
```

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

COBCVTRPQ
LIS

COBCVTDQ
LIS

COBCANCEL
LIS

COBCVTQP
LIS

COBACCTIM
LIS

COBCVTPQ
LIS

COBCVTRFQ
LIS

COBCVTQF
LIS

COBCVTRQP
LIS

COBCALL
LIS

COBCVTFQ
LIS

COBACCEPT
LIS

COBCVTRDQ
LIS

COBCVTQD
LIS