D 8

```
LLL                 IIIIIIIII     BBBBBBBBBBBB     RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
LLL                 IIIIIIIII     BBBBBBBBBBBB     RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
LLL                 IIIIIIIII     BBBBBBBBBBBB     RRRRRRRRRRRR     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        BBBBBBBBBBBB     RRRRRRRRRRRR           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
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
```

FILEID**LIBCVTMAC

```
LL            IIIIII    BBBBBBBB      CCCCCCCC  VV      VV  TTTTTTTTTT  MM      MM    AAAAAA      CCCCCCCC
LL            IIIIII    BBBBBBBB      CCCCCCCC  VV      VV  TTTTTTTTTT  MM      MM    AAAAAA      CCCCCCCC
LL              II      BB      BB  CC          VV      VV      TT      MMMM  MMMM  AA      AA  CC
LL              II      BB      BB  CC          VV      VV      TT      MMMM  MMMM  AA      AA  CC
LL              II      BB      BB  CC          VV      VV      TT      MM  MM  MM  AA      AA  CC
LL              II      BB      BB  CC          VV      VV      TT      MM  MM  MM  AA      AA  CC
LL              II      BBBBBBBB    CC          VV      VV      TT      MM      MM  AA      AA  CC
LL              II      BBBBBBBB    CC          VV      VV      TT      MM      MM  AA      AA  CC
LL              II      BB      BB  CC          VV      VV      TT      MM      MM  AAAAAAAAAA  CC
LL              II      BB      BB  CC          VV      VV      TT      MM      MM  AAAAAAAAAA  CC
LL              II      BB      BB  CC            VV  VV        TT      MM      MM  AA      AA  CC            ::
LL              II      BB      BB  CC            VV  VV        TT      MM      MM  AA      AA  CC            ::
LLLLLLLLLL    IIIIII    BBBBBBBB      CCCCCCCC      VV          TT      MM      MM  AA      AA    CCCCCCCC    ::
LLLLLLLLLL    IIIIII    BBBBBBBB      CCCCCCCC      VV          TT      MM      MM  AA      AA    CCCCCCCC    ::

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            .TITLE  LIB$CVTMAC          MACRO-32 support for LIB$CVT_DX_DX
0000            .IDENT  /1-005/             ; File: LIBCVTMAC.MAR  EDIT: PDG1005
0000
0000    ;
0000    ;****************************************************************************
0000    ;*                                                                          *
0000    ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000    ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000    ;*  ALL RIGHTS RESERVED.                                                    *
0000    ;*                                                                          *
0000    ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000    ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000    ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000    ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000    ;*  TRANSFERRED.                                                            *
0000    ;*                                                                          *
0000    ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000    ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    ;*  CORPORATION.                                                            *
0000    ;*                                                                          *
0000    ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000    ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000    ;*                                                                          *
0000    ;*                                                                          *
0000    ;****************************************************************************
0000
0000    ;++
0000    ; FACILITY: Run-time library.
0000    ;
0000    ; ABSTRACT:
0000    ;       This module contains the MACRO-32 support routines for LIB$CVT_DX_DX
0000    ;       routine.  All of the entry points are JSB entry points.
0000    ;
0000    ; ENVIRONMENT:
0000    ;       Some entry points require the Caller to have LIB$EMULATE or the
0000    ;       machine have the full instruction set of VAX-11 architecture.
0000    ;       Note that this module is for specific support of LIB$CVT_DX_DX routine,
0000    ;       hence it has no  RTL "globally defined" entry point, therefore this
0000    ;       module assumes that the caller has IV, FU, DV bits in PSW turned on.
0000    ;
0000    ; AUTHOR:
0000    ;       AUTHORS :  FAROKH MORSHED, PETER GILBERT  7-FEB-81, VERSION 1, VMS V3.0.
0000    ;
0000    ; MODIFIED BY:
0000    ;
0000    ; VERSION: 1
0000    ; 1001  Original.
0000    ; 1002  Make references to LIB$SIGNAL General mode addressing.  FM 30-OCT-81.
0000    ; 1003  Fix LIB$$CVT_CVTRHQ_R1 to round correctly.  FM 1-Mar-83
0000    ; 1004  Fix the bug that a zero integer to float does not look for a zero
0000    ;       value.  FM 13-Jan-84
0000    ; 1005  Fix bug in LIB$$CVT_CVTRHO_R1 in code avoiding integer overflow.
0000    ;       PDG 10-Jul-1984
0000    ;--
0000    ;
```

```
    0000          .SBTTL  DECLARATIONS
    0000
    0000  ;
    0000  ; PSECT DECLARATIONS:
    0000  ;
00000000          .PSECT  _LIB$CODE       PIC, SHR, LONG, EXE, NOWRT
    0000  ;
    0000  ; EXTERNAL ROUTINES
    0000  ;
    0000          .EXTERNAL       LIB$SIGNAL
    0000  ;
    0000  ;       MACRO
    0000  ;
    0000          $SSDEF
    0000
```

```
                 0000          .SBTTL  LIB$$CVT_CVTLB_R1
                 0000
                 0000  ;++
                 0000  ; FUNCTIONAL DESCRIPTION:
                 0000  ;       Convert LONGWORD to BYTE
                 0000  ;
                 0000  ;
                 0000  ; CALLING SEQUENCE:
                 0000  ;       LIB$$CVT_CVTLB_R1 ( long.rl.r, byte.wb.r )
                 0000  ;       This is a JSB entry point.
                 0000  ;
                 0000  ;
                 0000  ; FORMAL PARAMETERS:
                 0000  ;       R0 --> long     R1 --> byte
                 0000  ;
                 0000  ;
                 0000  ; IMPLICIT INPUTS:
                 0000  ;       NONE
                 0000  ;
                 0000  ;
                 0000  ; IMPLICIT OUTPUTS:
                 0000  ;       NONE
                 0000  ;
                 0000  ;
                 0000  ; COMPLETION CODES:
                 0000  ;       NONE
                 0000  ;
                 0000  ; SIDE EFFECTS:
                 0000  ;
                 0000  ;
                 0000  ;--
                 0000
                 0000  LIB$$CVT_CVTLB_R1 ::
  61   60   F6   0000          CVTLB   (R0), (R1)              ;Convert LONGWORD to BYTE
            05   0003          RSB
```

```
               0004          .SBTTL  LIB$$CVT_CVTLW_R1
               0004
               0004   ;++
               0004   ; FUNCTIONAL DESCRIPTION:
               0004          Convert LONGWORD to WORD.
               0004   ;
               0004   ;
               0004   ; CALLING SEQUENCE:
               0004          LIB$$CVT_CVTLW_R1 ( long.rl.r, word.ww.r )
               0004          This is a JSB entry point.
               0004   ;
               0004   ;
               0004   ; FORMAL PARAMETERS:
               0004          R0 --> long      R1 --> word
               0004   ;
               0004   ;
               0004   ; IMPLICIT INPUTS:
               0004          NONE
               0004   ;
               0004   ;
               0004   ; IMPLICIT OUTPUTS:
               0004          NONE
               0004   ;
               0004   ;
               0004   ; COMPLETION CODES:
               0004          NONE
               0004   ;
               0004   ; SIDE EFFECTS:
               0004   ;
               0004   ;
               0004   ;--
               0004
               0004   LIB$$CVT_CVTLW_R1 ::
 61   60   F7  0004          CVTLW   (R0), (R1)            ;Convert LONGWORD to WORD
           05  0007          RSB
```

```
                0008        .SBTTL  LIB$$CVT_CVTLH_R1
                0008
                0008  ;++
                0008  ; FUNCTIONAL DESCRIPTION:
                0008  ;       Convert LONGWORD to H_FLOATING.
                0008  ;
                0008  ;
                0008  ; CALLING SEQUENCE:
                0008  ;       LIB$$CVT_CVTLH_R1 ( long.rl.r, hfloat.wh.r )
                0008  ;       This is a JSB entry point.
                0008  ;
                0008  ;
                0008  ; FORMAL PARAMETERS:
                0008  ;       R0 --> long       R1 --> hfloat
                0008  ;
                0008  ;
                0008  ; IMPLICIT INPUTS:
                0008  ;       NONE
                0008  ;
                0008  ;
                0008  ; IMPLICIT OUTPUTS:
                0008  ;       NONE
                0008  ;
                0008  ;
                0008  ; COMPLETION CODES:
                0008  ;       NONE
                0008  ;
                0008  ; SIDE EFFECTS:
                0008  ;       OPCODE RESERVED TO DIGITAL error is possible, see ENVIRONMENT.
                0008  ;
                0008  ;--
                0008
                0008  LIB$$CVT_CVTLH_R1 ::
   61   60 6EFD 0008        CVTLH   (R0), (R1)                    ;Convert LONGWORD to H_FLOATING
             05 000C        RSB
                000D
```

```
                     000D        .SBTTL  LIB$$CVT_CVTROUF_R1
                     000D
                     000D  ;++
                     000D  ; FUNCTIONAL DESCRIPTION:
                     000D  ;         Convert unsigned OCTAWORD to FLOAT.
                     000D  ;
                     000D  ;
                     000D  ; CALLING SEQUENCE:
                     000D  ;         LIB$$CVT_CVTROUF_R1 ( octa.ro.r, float.wf.r )
                     000D  ;         This is a JSB entry point.
                     000D  ;
                     000D  ;
                     000D  ; FORMAL PARAMETERS:
                     000D  ;         R0 --> octa      R1 --> float
                     000D  ;
                     000D  ;
                     000D  ; IMPLICIT INPUTS:
                     000D  ;         NONE
                     000D  ;
                     000D  ;
                     000D  ; IMPLICIT OUTPUTS:
                     000D  ;         NONE
                     000D  ;
                     000D  ;
                     000D  ; COMPLETION CODES:
                     000D  ;         NONE
                     000D  ;
                     000D  ; SIDE EFFECTS:
                     000D  ;         OPCODE RESERVED TO DIGITAL error is possible, see ENVIRONMENT.
                     000D  ;
                     000D  ;--
                     000D
                     000D  ;
                     000D  ;           DEFINE SOME CONSTANTS
                     000D  ;
          00000011   000D  FDUMMY: .BLKL          1
          00007080   0011          .F_FLOATING    79228162514264337593543950336     ;2**96
          00006080   0015          .F_FLOATING    18446744073709551616              ;2**64
          00005080   0019  F32:    .F_FLOATING    4294967296                        ;2**32
          00004080   001D          .F_FLOATING    1                                 ;
                     0021  ;
                     0021  ;         FIND THE FLOATING ATOMIC DATA TYPE BY MULTIPLYING EACH LONGWORD
                     0021  ;         OF OCTAWORD BY AN APPROPRIATE CONSTANT.
                     0021  ;
                     0021  LIB$$CVT_CVTROUF_R1 ::
            0F   BB  0021          PUSHR   #^M<R0,R1,R2,R3>
            61   D4  0023          CLRF    (R1)                  ;Initialize destination.
       52   04   D0  0025          MOVL    #4, R2                ;Set up the loop counter for 4 times
       53   60   4E  0028  10$:    CVTLF   (R0), R3              ;Convert the next LONGWORD to floating
            06   14  002B          BGTR    13$                   ;This longword is positive
            0C   13  002D          BEQL    15$                   ;If zero, then just skip it
     53   E7 AF  40  002F          ADDF2   F32, R3               ;Negative, so add the difference
  53   D6 AF42   44  0033  13$:    MULF2   FDUMMY[R2], R3        ;Multiply the result by the constant
       61   53   40  0038          ADDF2   R3, (R1)              ;Add in the result to destination
            52   97  003B  15$:    DECB    R2                    ;One less time to go around in the loop
            08   13  003D          BEQL    20$                   ;Finished if counter is zero
            80   D5  003F          TSTL    (R0)+                 ;Next longword of OCTA
```

```
60   D5  0041   240         TSTL    (R0)                    ;Is it zero ?
F6   13  0043   241         BEQL    15$                     ;Yes, so ignore it
E1   11  0045   242         BRB     10$                     ;Loop to chomp some more on the OCTA
         0047   243 20$:
0F   BA  0047   244         POPR    #^M<R0,R1,R2,R3>
     05  0049   245         RSB
```

```
                          004A         .SBTTL  LIB$$CVT_CVTROUD_R1
                          004A
                          004A  ;++
                          004A  ; FUNCTIONAL DESCRIPTION:
                          004A  ;       Convert unsigned OCTAWORD to D_FLOATING.
                          004A  ;
                          004A  ;
                          004A  ; CALLING SEQUENCE:
                          004A  ;       LIB$$CVT_CVTROUD_R1 ( octa.ro.r, dfloat.wd.r )
                          004A  ;       This is a JSB entry point.
                          004A  ;
                          004A  ;
                          004A  ; FORMAL PARAMETERS:
                          004A  ;       R0 --> octa      R1 --> dfloat
                          004A  ;
                          004A  ;
                          004A  ; IMPLICIT INPUTS:
                          004A  ;       NONE
                          004A  ;
                          004A  ;
                          004A  ; IMPLICIT OUTPUTS:
                          004A  ;       NONE
                          004A  ;
                          004A  ;
                          004A  ; COMPLETION CODES:
                          004A  ;       NONE
                          004A  ;
                          004A  ; SIDE EFFECTS:
                          004A  ;       OPCODE RESERVED TO DIGITAL error is possible, see ENVIRONMENT.
                          004A  ;
                          004A  ;--
                          004A
                          004A  ;
                          004A  ;         DEFINE SOME CONSTANTS
                          004A  ;
         00000052         004A  DDUMMY: .BLKL          2
00000000 00007080         0052          .D_FLOATING    79228162514264337593543950336   ;2**96
00000000 00006080         005A          .D_FLOATING    18446744073709551616            ;2**64
00000000 00005080         0062  D32:    .D_FLOATING    4294967296                      ;2**32
00000000 00004080         006A          .D_FLOATING    1                               ;
                          0072  ;
                          0072  ;        FIND THE FLOATING ATOMIC DATA TYPE BY MULTIPLYING EACH LONGWORD
                          0072  ;        OF OCTAWORD BY AN APPROPRIATE CONSTANT.
                          0072  ;
                          0072  LIB$$CVT_CVTROUD_R1 ::
             1F    BB     0072          PUSHR   #^M<R0,R1,R2,R3,R4>
             61    7C     0074          CLRD    (R1)                ;Initialize destination
       52    04    D0     0076          MOVL    #4, R2              ;Set up the loop counter for 4 times
       53    60    6E     0079  10$:    CVTLD   (R0), R3            ;Convert the next LONGWORD to floating
             06    14     007C          BGTR    13$                 ;This longword is positive
             0C    13     007E          BEQL    15$                 ;If zero, then just skip it
    53    DF AF    60     0080          ADDD2   D32, R3             ;Negative, so add the difference
53    C2 AF42      64     0084  13$:    MULD2   DDUMMY[R2], R3      ;Multiply the result by the constant
       61    53    60     0089          ADDD2   R3, (R1)            ;Add in the result to destination
             52    97     008C  15$:    DECB    R2                  ;One less time to go around in the loop
             08    13     008E          BEQL    20$                 ;Finished if counter is zero
             80    D5     0090          TSTL    (R0)+               ;Next longword of OCTA
```

```
60    D5  0092   304          TSTL    (R0)                      ;Is it zero
F6    13  0094   305          BEQL    15$                       ;Yes, so ignore it
E1    11  0096   306          BRB     10$                       ;Loop to chomp some more on the OCTA
          0098   307  20$:
1F    BA  0098   308          POPR    #^M<R0,R1,R2,R3,R4>
      05  009A   309          RSB
```

```
                              009B          .SBTTL  LIB$$CVT_CVTROUG_R1
                              009B
                              009B  ;++
                              009B  ; FUNCTIONAL DESCRIPTION:
                              009B  ;       Convert unsigned OCTAWORD to G_FLOATING
                              009B  ;
                              009B  ;
                              009B  ; CALLING SEQUENCE:
                              009B  ;       LIB$$CVT_CVTROUG_R1 ( octa.ro.r, gfloat.wg.r )
                              009B  ;       This is a JSB entry point.
                              009B  ;
                              009B  ;
                              009B  ; FORMAL PARAMETERS:
                              009B  ;       R0 --> octa      R1 --> gfloat
                              009B  ;
                              009B  ;
                              009B  ; IMPLICIT INPUTS:
                              009B  ;       NONE
                              009B  ;
                              009B  ;
                              009B  ; IMPLICIT OUTPUTS:
                              009B  ;       NONE
                              009B  ;
                              009B  ;
                              009B  ; COMPLETION CODES:
                              009B  ;       NONE
                              009B  ;
                              009B  ; SIDE EFFECTS:
                              009B  ;       OPCODE RESERVED TO DIGITAL error is possible, see ENVIRONMENT.
                              009B  ;
                              009B  ;--
                              009B
                              009B  ;
                              009B  ;         DEFINE SOME CONSTANTS
                              009B  ;
               000000A3       009B  GDUMMY: .BLKL           2
      00000000 00004610       00A3          .G_FLOATING     79228162514264337593543950336;2**96
      00000000 00004410       00AB          .G_FLOATING     18446744073709551616;2**64
      00000000 00004210       00B3  G32:    .G_FLOATING     4294967296;2**32
      00000000 00004010       00BB          .G_FLOATING     1;
                              00C3  ;
                              00C3  ;        FIND THE FLOATING ATOMIC DATA TYPE BY MULTIPLYING EACH LONGWORD
                              00C3  ;        OF OCTAWORD BY AN APPROPRIATE CONSTANT.
                              00C3  ;
                              00C3  LIB$$CVT_CVTROUG_R1 ::
                  1F     BB   00C3          PUSHR   #^M<R0,R1,R2,R3,R4>
                  61     7C   00C5          CLRG    (R1)                    ;Initialize destination
              52  04     D0   00C7          MOVL    #4, R2                  ;Set up the loop counter for 4 times
              53  60   4EFD   00CA  10$:    CVTLG   (R0), R3                ;Convert the next LONGWORD to floating
                  07     14   00CE          BGTR    13$                     ;This longword is positive
                  0F     13   00D0          BEQL    15$                     ;If zero, then just skip it
          53   DD AF   40FD   00D2          ADDG2   G32, R3                 ;Negative, so add the difference
      53   BF AF42     44FD   00D7  13$:    MULG2   GDUMMY[R2], R3          ;Multiply the result by the constant
              61  53   40FD   00DD          ADDG2   R3, (R1)                ;Add in the result to destination
                  52     97   00E1  15$:    DECB    R2                      ;One less time to go around in the loop
                  08     13   00E3          BEQL    20$                     ;Finished if counter is zero
                  80     D5   00E5          TSTL    (R0)+                   ;Next longword of OCTA
```

```
60   D5  00E7   368              TSTL    (R0)                    ;Is it zero ?
F6   13  00E9   369              BEQL    15$                     ;Yes, so ignore it
DD   11  00EB   370              BRB     10$                     ;Loop to chomp some more on the OCTA
         00ED   371  20$:
1F   BA  00ED   372              POPR    #^M<R0,R1,R2,R3,R4>
     05  00EF   373              RSB
```

```
                      00F0          .SBTTL  LIB$$CVT_CVTROUH_R1
                      00F0
                      00F0  ;++
                      00F0  ; FUNCTIONAL DESCRIPTION:
                      00F0  ;       Convert unsigned OCTAWORD to H_FLOATING rounded.
                      00F0  ;
                      00F0  ;
                      00F0  ; CALLING SEQUENCE:
                      00F0  ;       LIB$$CVT_CVTROUH_R1 ( octa.ro.r, hfloat.wh.r )
                      00F0  ;       This is a JSB entry point.
                      00F0  ;
                      00F0  ;
                      00F0  ; FORMAL PARAMETERS:
                      00F0  ;       R0 --> octa     R1 --> hfloat
                      00F0  ;
                      00F0  ;
                      00F0  ; IMPLICIT INPUTS:
                      00F0  ;       NONE
                      00F0  ;
                      00F0  ;
                      00F0  ; IMPLICIT OUTPUTS:
                      00F0  ;       NONE
                      00F0  ;
                      00F0  ;
                      00F0  ; COMPLETION CODES:
                      00F0  ;       NONE
                      00F0  ;
                      00F0  ; SIDE EFFECTS:
                      00F0  ;
                      00F0  ;--
                      00F0
                      00F0  LIB$$CVT_CVTROUH_R1 ::
            0F   BB   00F0          PUSHR   #^M<R0,R1,R2,R3>
            60   D5   00F2          TSTL    (R0)                    ;If OU is zero, return zero.
            1A   12   00F4          BNEQ    5$                      ;....
         04 A0   D5   00F6          TSTL    4(R0)                   ;....
            15   12   00F9          BNEQ    5$                      ;....
         08 A0   D5   00FB          TSTL    8(R0)                   ;....
            10   12   00FE          BNEQ    5$                      ;....
         0C A0   D5   0100          TSTL    12(R0)                  ;....
            0B   12   0103          BNEQ    5$                      ;....
            81   D4   0105          CLRL    (R1)+                   ;Return 0 in H.
            81   D4   0107          CLRL    (R1)+                   ;....
            81   D4   0109          CLRL    (R1)+                   ;....
            61   D4   010B          CLRL    (R1)                    ;....
          0079   31   010D          BRW     50$                     ;Go to RSB back to caller.
                      0110  5$:
    5E      12   C2   0110          SUBL2   #18, SP                 ;Make room for nomalized OU.
                      0113  ;
                      0113  ; Move source to top of stack to normalize it
                      0113  ;
    6E      60   7D   0113          MOVQ    (R0), (SP)              ;Put OU in here to be
 08 AE   08 A0   7D   0116          MOVQ    8(R0), 8(SP)            ;       worked on.
         10 AE   B4   011B          CLRW    16(SP)                  ;We need this extra word for ADWC.
                      011E  ;
                      011E  ; Initialize some locations
                      011E  ;
```

```
                   52   D4 011E          CLRL    R2                   ;R2 will contain number of shifts.
                   53   D4 0120          CLRL    R3                   ;Temporary.
                   20   B9 0122          BICPSW  #^X20                ;Turn off IV so we can ASHQ
                           0124  ;
                           0124  ; This loop will shift OU left until MSB is a one.
                           0124  ;
                           0124  10$:
                   52   D7 0124          DECL    R2                   ;Count number of shifts.
                 0063   30 0126          BSBW    SHIFT_OU_LEFT        ;Shift OU left one time.
                0F AE   95 0129          TSTB    15(SP)               ;Is MSB of normalized OU set ?
                   F6   18 012C          BGEQ    10$                  ;No, loop again.
                           012E  ;
                           012E  ; Round the OU if needed
                           012E  ;
          1D 6E    0E   E1 012E          BBC     #14, (SP), 30$       ;If the bit beyond LSB of H clear no
                           0132                                       ;        rounding is needed.
          19 6E    0F   E3 0132          BBCS    #15, (SP), 30$       ;If this bit is clear we just need
                           0136                                       ;        to set it and it is rounded.
       01 AE    80 8F   8A 0136          BICB2   #^X80, 1(SP)         ;Painfully round this bit pattern.
          02 AE    01   C0 013B          ADDL2   #1, 2(SP)            ;.....
          06 AE    00   D8 013F          ADWC    #0, 6(SP)            ;.....
          0A AE    00   D8 0143          ADWC    #0, 10(SP)           ;.....
          0E AE    00   D8 0147          ADWC    #0, 14(SP)           ;.....
                           014B  ;
                           014B  ; Normalize this bit pattern
                           014B  ;
             03 10 AE   E8 014B          BLBS    16(SP), 40$          ;If this bit is on, it means that we
                           014F                                       ;        had an overflow on the last
                           014F                                       ;        word, so it is already
                           014F                                       ;        normalized.
                           014F  30$:
                 003A   30 014F          BSBW    SHIFT_OU_LEFT        ;Normalize it.
                           0152  ;
                           0152  ; Put what we have cooked up in the H destination
                           0152  ;
                           0152  40$:
    52   00000080 8F    C0 0152          ADDL2   #128, R2             ;exp = length of OU (128) - #of shifts.
                61 52   B0 0159          MOVW    R2, (R1)             ;Put in exponent.
       01 A1    40 8F   88 015C          BISB2   #^X40, 1(R1)         ;Set the excess bit.
       02 A1    0E AE   B0 0161          MOVW    14(SP), 2(R1)        ;Put in the fraction.
       04 A1    0C AE   B0 0166          MOVW    12(SP), 4(R1)        ;......
       06 A1    0A AE   B0 016B          MOVW    10(SP), 6(R1)        ;......
       08 A1    08 AE   B0 0170          MOVW    8(SP), 8(R1)         ;......
       0A A1    06 AE   B0 0175          MOVW    6(SP), 10(R1)        ;......
       0C A1    04 AE   B0 017A          MOVW    4(SP), 12(R1)        ;......
       0E A1    02 AE   B0 017F          MOVW    2(SP), 14(R1)        ;......
                   20   B8 0184          BISPSW  #^X20                ;Turn IV back on.
                5E 12   C0 0186          ADDL2   #18, SP              ;Restore SP
                   0F   BA 0189  50$:    POPR    #^M<R0,R1,R2,R3>
                   05   05 018B          RSB                          ;RSB back to caller.
                           018C  ;
                           018C  ; Subroutine to shift OU one to the left
                           018C  ;
                           018C  SHIFT_OU_LEFT:
                   53   94 018C          CLRB    R3                   ;Initialize R3
                0B AE   95 018E          TSTB    11(SP)               ;Is MSB of low quad set.
                   02   18 0191          BGEQ    10$                  ;No.
```

```
                         53   96 0193   489         INCB    R3                      ;Yes, remember it.
                                 0195   490 10$:
     04 AE    04 AE      01   79 0195   491         ASHQ    #1, 4(SP), 4(SP)        ;Shift low quad one time.
     0C AE    0C AE      01   79 019B   492         ASHQ    #1, 12(SP), 12(SP)      ;Shift high quad one time.
   0C AE   01    00      53   F0 01A1   493         INSV    R3, #0, #1, 12(SP)      ;If MSB of low quad was set then set
                                 01A7   494                                         ;LSB of high quad.
                              05 01A7   495         RSB                             ;Return to main routine.
```

```
                          01A8          .SBTTL  LIB$$CVT_CVTRDQ_R1
                          01A8
                          01A8  ;++
                          01A8  ; FUNCTIONAL DESCRIPTION:
                          01A8  ;       Convert D_FLOATING to QUADWORD rounded.
                          01A8  ;
                          01A8  ;
                          01A8  ; CALLING SEQUENCE:
                          01A8  ;       LIB$$CVT_CVTRDQ_R1 ( dfloat.rd.r, quad.wq.r )
                          01A8  ;       This is a JSB entry point.
                          01A8  ;
                          01A8  ;
                          01A8  ; FORMAL PARAMETERS:
                          01A8  ;       R0 --> dfloat   R1 --> QUADWORD
                          01A8  ;
                          01A8  ;
                          01A8  ; IMPLICIT INPUTS:
                          01A8  ;       NONE
                          01A8  ;
                          01A8  ;
                          01A8  ; IMPLICIT OUTPUTS:
                          01A8  ;       NONE
                          01A8  ;
                          01A8  ;
                          01A8  ; COMPLETION CODES:
                          01A8  ;       NONE
                          01A8  ;
                          01A8  ; SIDE EFFECTS:
                          01A8  ;       NONE
                          01A8  ;
                          01A8  ;--
                          01A8
           00000007       01A8  D_OFF   = 7                                     ; Offset to exponent (double)
                          01A8
                          01A8  LIB$$CVT_CVTRDQ_R1 ::
              0F    BB    01A8          PUSHR   #^M<R0,R1,R2,R3>                ; Save these registers
              61    7C    01AA          CLRD    (R1)                            ; Initialize result to zero
          52  60    70    01AC          MOVD    (R0), R2                        ; Grab source (reserved operand?)
              24    13    01AF          BEQL    20$                             ; Branch if result is zero
  52    1000 8F    A2    01B1          SUBW2   #32@D_OFF, R2                   ; Scale down by 2**32
   04 A1     52    6B    01B6          CVTRDL  R2, 4(R1)                       ; Store high-order longword of result
  52    1000 8F    A0    01BA          ADDW2   #32@D_OFF, R2                   ; Scale up by 2**32
     7E   04 A1    6E    01BF          CVTLD   4(R1), -(SP)                    ; Convert long back to double
              05    13    01C3          BEQL    10$                             ; Skip if zero
  6E    1000 8F    A0    01C5          ADDW2   #32@D_OFF, (SP)                 ; Scale up by 2**32
          52  8E    62    01CA  10$:    SUBD2   (SP)+, R2                       ; Subtract high-order longword
          61  52    6B    01CD          CVTRDL  R2, (R1)                        ; Convert rounded remainder to long
              03    18    01D0          BGEQ    20$                             ; Skip if positive
           04 A1    D7    01D2          DECL    4(R1)                           ; Sign-extend low-order longword
              0F    BA    01D5  20$:    POPR    #^M<R0,R1,R2,R3>                ; Restore these registers
              05          01D7          RSB
```

```
                    01D8          .SBTTL  LIB$$CVT_CVTDH_R1
                    01D8
                    01D8    ;++
                    01D8    ; FUNCTIONAL DESCRIPTION:
                    01D8    ;       Convert D_FLOATING to H_FLOATING.
                    01D8    ;
                    01D8    ;
                    01D8    ; CALLING SEQUENCE:
                    01D8    ;       LIB$$CVT_CVTDH_R1 ( dfloat.rd.r, hfloat.wh.r )
                    01D8    ;       This is a JSB entry point.
                    01D8    ;
                    01D8    ;
                    01D8    ; FORMAL PARAMETERS:
                    01D8    ;       R0 --> dfloat   R1 --> hfloat
                    01D8    ;
                    01D8    ;
                    01D8    ; IMPLICIT INPUTS:
                    01D8    ;       NONE
                    01D8    ;
                    01D8    ;
                    01D8    ; IMPLICIT OUTPUTS:
                    01D8    ;       NONE
                    01D8    ;
                    01D8    ;
                    01D8    ; COMPLETION CODES:
                    01D8    ;       NONE
                    01D8    ;
                    01D8    ; SIDE EFFECTS:
                    01D8    ;       OPCODE RESERVED TO DIGITAL error is possible, see ENVIRONMENT.
                    01D8    ;
                    01D8    ;--
                    01D8
                    01D8    LIB$$CVT_CVTDH_R1 ::
61   60 32FD        01D8            CVTDH   (R0), (R1)              ;Convert D_FLOATING to H_FLOATING
             05     01DC            RSB
```

```
                 01DD            .SBTTL  LIB$$CVT_CVTRHL_R1
                 01DD
                 01DD     ;++
                 01DD     ; FUNCTIONAL DESCRIPTION:
                 01DD     ;       Convert H_FLOATING to LONGWORD rounded.
                 01DD     ;
                 01DD     ;
                 01DD     ; CALLING SEQUENCE:
                 01DD     ;       LIB$$CVT_CVTRHL_R1 ( hfloat.rh.r, long.wl.r )
                 01DD     ;       This is a JSB entry point.
                 01DD     ;
                 01DD     ;
                 01DD     ; FORMAL PARAMETERS:
                 01DD     ;       R0 --> hfloat   R1 --> long
                 01DD     ;
                 01DD     ;
                 01DD     ; IMPLICIT INPUTS:
                 01DD     ;       NONE
                 01DD     ;
                 01DD     ;
                 01DD     ; IMPLICIT OUTPUTS:
                 01DD     ;       NONE
                 01DD     ;
                 01DD     ;
                 01DD     ; COMPLETION CODES:
                 01DD     ;       NONE
                 01DD     ;
                 01DD     ; SIDE EFFECTS:
                 01DD     ;       OPCODE RESERVED TO DIGITAL error is possible, see ENVIRONMENT.
                 01DD     ;
                 01DD     ;--
                 01DD     LIB$$CVT_CVTRHL_R1 ::
  61   60 6BFD   01DD            CVTRHL  (R0), (R1)               ;Convert H_FLOATING to LONGWORD rounded
              05 01E1            RSB
```

```
                            01E2          .SBTTL  LIB$$CVT_CVTRHQ_R1
                            01E2    ;++
                            01E2    ; FUNCTIONAL DESCRIPTION:
                            01E2    ;       Convert H_FLOATING to QUADWORD rounded.
                            01E2    ;
                            01E2    ;
                            01E2    ; CALLING SEQUENCE:
                            01E2    ;       LIB$$CVT_CVTRHQ_R1 ( hfloat.rh.r, quad.wq.r )
                            01E2    ;       This is a JSB entry point.
                            01E2    ;
                            01E2    ;
                            01E2    ; FORMAL PARAMETERS:
                            01E2    ;       R0 --> hfloat   R1 --> quad
                            01E2    ;
                            01E2    ;
                            01E2    ; IMPLICIT INPUTS:
                            01E2    ;       NONE
                            01E2    ;
                            01E2    ;
                            01E2    ; IMPLICIT OUTPUTS:
                            01E2    ;       NONE
                            01E2    ;
                            01E2    ;
                            01E2    ; COMPLETION CODES:
                            01E2    ;       NONE
                            01E2    ;
                            01E2    ; SIDE EFFECTS:
                            01E2    ;
                            01E2    ;
                            01E2    ;--
                            01E2    LIB$$CVT_CVTRHQ_R1 ::
               0F    BB     01E2            PUSHR   #^M<R0,R1,R2,R3>        ; Save these registers
               61    7C     01E4            CLRQ    (R1)                    ; Initialize result to zero
               53    D4     01E6            CLRL    R3                      ; Initialize rounding bit
                            01E8    ;
                            01E8    ;        Find the exponent
                            01E8    ;
   52   60   0F    00    EF     01E8            EXTZV   #0, #15, (R0), R2       ; Extract the exponent
               0D    13     01ED            BEQL    5$                      ; Call the number zero if expo. is zero
   52   00004000 8F    C2     01EF            SUBL2   #^X4000, R2             ; Take out the excess
               04    19     01F6            BLSS    5$                      ; If negative then Quad is zero
               05    14     01F8            BGTR    10$                     ; Is exponent zero ?
               61    D6     01FA            INCL    (R1)                    ; Yes, call it a one
                            01FC    5$:
             007D    31     01FC            BRW     50$                     ; Go to the finish line
                            01FF    ;
                            01FF    ;        Check for overflow.  If exponent is 63, find rounding bit before we
                            01FF    ;        lose it.
                            01FF    ;
                            01FF    10$:
          52   3F    C2     01FF            SUBL2   #63, R2                 ; Bring '.' to right of bit 0
               1E    19     0202            BLSS    20$                     ; If exponent < 63, skip this
               0F    14     0204            BGTR    15$                     ; If exponent > 63, we have IV
   53   08 A0   01    01    EF     0206            EXTZV   #1, #1, 8(R0), R3       ; Exponent is 63, R3 is the round bit
   53   80000000 8F    CA     020C            BICL2   #^X80000000, R3         ; Turn on the 31st bit of R3 to flag
                            0213                                            ; that the round bit has been determined
               0D    11     0213            BRB     20$                     ; Cont.
```

```
                              0215  15$:
        0000047C 8F      DD   0215          PUSHL   #SS$_INTOVF             ; Signal IV
       00000000'GF 01    FB   021B          CALLS   #1, G^LIB$SIGNAL        ;
                              0222  ;
                              0222  ;       Move the fraction to the QUAD, and put in a positive sign and MSB
                              0222  ;
                              0222  20$:
           61    08 A0   B0   0222          MOVW    8(R0), (R1)             ; Move the fraction to QUAD
        02 A1    06 A0   B0   0226          MOVW    6(R0), 2(R1)            ;
        04 A1    04 A0   B0   022B          MOVW    4(R0), 4(R1)            ;
        06 A1    02 A0   B0   0230          MOVW    2(R0), 6(R1)            ;
     61    61    FE 8F   79   0235          ASHQ    #-2, (R1), (R1)         ; Make room for sign and MSB
        07 A1    C0 8F   8A   023A          BICB2   #^XC0, 7(R1)            ; Make sure two high bits are off
        07 A1    40 8F   88   023F          BISB2   #^X40, 7(R1)            ; Sign is '+', MSB is 1
                              0244  ;
                              0244  ;       Find rounding bit if not already done
                              0244  ;
                     53  D5   0244          TSTL    R3                      ; Is rounding bit already found ?
                     0A  19   0246          BLSS    30$                     ; Yes
              53     52  CE   0248          MNEGL   R2, R3                  ; R3 will contain rounding bit
                     53  D7   024B          DECL    R3                      ; Look at the right of decimal point
   53   61    01     53  EF   024D          EXTZV   R3, #1, (R1), R3        ; This is the rounding bit
                              0252  ;
                              0252  ;       Shift the QAUD R2 times
                              0252  ;
                              0252  30$:
        61    61     52  79   0252          ASHQ    R2, (R1), (R1)          ; Shift the QUAD R2 times
                     53  95   0256          TSTB    R3                      ; Do we need to round
                     0B  13   0258          BEQL    40$                     ; No
              52     61  D0   025A          MOVL    (R1), R2                ; Do the rounding
           61    01  A2  9E   025D          MOVAB   1(R2), (R1)             ; ...
        04 A1    00      D8   0261          ADWC    #0, 4(R1)               ; ...
                     60  B5   0265  40$:    TSTW    (R0)                    ; Is the H negative ?
                     13  14   0267          BGTR    50$                     ; No
              61     61  D2   0269          MCOML   (R1), (R1)              ; Negate the QUAD
        04 A1    04  A1  D2   026C          MCOML   4(R1), 4(R1)            ; ...
              50     61  D0   0271          MOVL    (R1), R0                ; ...
           61    01  A0  9E   0274          MOVAB   1(R0), (R1)             ; ...
        04 A1    00      D8   0278          ADWC    #0, 4(R1)               ; ...
                              027C  50$:
                     0F  BA   027C          POPR    #^M<R0,R1,R2,R3>        ; Restore these registers
                         05   027E          RSB
```

```
                         027F        .SBTTL  LIB$$CVT_CVTRHO_R1
                         027F   ;++
                         027F   ; FUNCTIONAL DESCRIPTION:
                         027F   ;       Convert H_FLOATING to OCTAWORD rounded.
                         027F   ;
                         027F   ;
                         027F   ; CALLING SEQUENCE:
                         027F   ;       LIB$$CVT_CVTRHO_R1 ( hfloat.rh.r, octa.wo.r )
                         027F   ;       This is a JSB entry point.
                         027F   ;
                         027F   ;
                         027F   ; FORMAL PARAMETERS:
                         027F   ;       R0 --> hfloat   R1 --> octa
                         027F   ;
                         027F   ;
                         027F   ; IMPLICIT INPUTS:
                         027F   ;       NONE
                         027F   ;
                         027F   ;
                         027F   ; IMPLICIT OUTPUTS:
                         027F   ;       NONE
                         027F   ;
                         027F   ;
                         027F   ; COMPLETION CODES:
                         027F   ;       NONE
                         027F   ;
                         027F   ; SIDE EFFECTS:
                         027F   ;       OPCODE RESERVED TO DIGITAL error is possible, see ENVIRONMENT.
                         027F   ;
                         027F   ;--
          00000004       027F   O_LEN   = 4                ; Length in longwords (octaword)
                         027F
                         027F   LIB$$CVT_CVTRHO_R1 ::
                         027F           ;
                         027F           ; Initialization
                         027F           ;
             3F   BB     027F           PUSHR   #^M<R0,R1,R2,R3,R4,R5>  ; Save these registers
             61   7C     0281           CLRQ    (R1)                    ; Initialize result to zero
          08 A1   7C     0283           CLRQ    8(R1)                   ; Initialize result to zero
       7E    60 70FD     0286           MOVH    (R0), -(SP)             ; Grab source (reserved operand?)
             43   13     028A           BEQL    30$                     ; Branch if result is zero
                         028C           ;
                         028C           ; The source is not zero; initialize other registers
                         028C           ;
       51    10   C0     028C           ADDL2   #O_LEN*4, R1            ; Address past octaword
       54    51   D0     028F           MOVL    R1, R4                  ; Save this address
       55    03   D0     0292           MOVL    #O_LEN-1, R5            ; Loop counter
                         0295           ;
                         0295   10$:    ; Loop for each longword of destination
                         0295           ;
  52   55    05   78     0295           ASHL    #5, R5, R2              ; Scaling amount (2**5 = 32)
       53    51   D0     0299           MOVL    R1, R3                  ; Address where we start decrementing
       6E    52   A2     029C           SUBW2   R2, (SP)                ; Scale down by 2**(32*n)
       71    6E 6BFD     029F           CVTRHL  (SP), -(R1)             ; Store high-order longword of result
       6E    52   A0     02A3           ADDW2   R2, (SP)                ; Scale up by 2**(32*n)
       7E    61 6EFD     02A6           CVTLH   (R1), -(SP)             ; Convert long back to double
             1C   13     02AA           BEQL    20$                     ; Skip if zero
```

```
               10      11 02AC          BRB     13$                  ; Jump into decrement loop
                          02AE          ;
                          02AE          ; If the longword was negative, we must sign-extend the result
                          02AE          ;
80000000 8F    63      D1 02AE  11$:    CMPL    (R3), #^X80000000    ; Compare with largest negative number
               05      12 02B5          BNEQ    12$                  ; If not equal, do a simple decrement
          83   63      D2 02B7          MCOML   (R3), (R3)+          ; Subtract one and consider negative
               04      11 02BA          BRB     14$                  ; Stay in loop, as result is negative
               83      D7 02BC  12$:    DECL    (R3)+                ; Decrement this longword
               05      18 02BE  13$:    BGEQ    15$                  ; Jump out of loop if positive
          54   53      D1 02C0  14$:    CMPL    R3, R4               ; See if we are done decrementing
               E9      1F 02C3          BLSSU   11$                  ; If not past octaword, keep going
                          02C5          ;
                          02C5          ; Scale up the longword (if not zero), and subtract from stack temp
                          02C5          ;
          6E   52      A0 02C5  15$:    ADDW2   R2, (SP)             ; Scale up by 2**(32*n)
          6E   8E    62FD 02C8  20$:    SUBH2   (SP)+, (SP)          ; Subtract high-order longword
                          02CC          ;
                          02CC          ; Decrease loop counter, and continue, if more longwords required
                          02CC          ;
            C6 55      F4 02CC          SOBGEQ  R5, 10$              ; Continue looping
                          02CF          ;
                          02CF  30$:    ; Clean up the stack, and restore registers
                          02CF          ;
               8E    7CFD 02CF          CLRH    (SP)+                ; Pop huge from the stack
               3F      BA 02D2          POPR    #^M<R0,R1,R2,R3,R4,R5> ; Restore these registers
                       05 02D4          RSB                          ; Return from subroutine
                          02D5
```

```
                      02D5              .SBTTL  LIB$$CVT_CVTHF_R1
                      02D5
                      02D5      ;++
                      02D5      ; FUNCTIONAL DESCRIPTION:
                      02D5      ;       Convert H_FLOATING to FLOATING
                      02D5      ;
                      02D5      ;
                      02D5      ; CALLING SEQUENCE:
                      02D5      ;       LIB$$CVT_CVTHF_R1 (hfloat.rh.r, float.wf.r )
                      02D5      ;       This is a JSB entry point.
                      02D5      ;
                      02D5      ;
                      02D5      ; FORMAL PARAMETERS:
                      02D5      ;       R0 --> hfloat   R1 --> float
                      02D5      ;
                      02D5      ;
                      02D5      ; IMPLICIT INPUTS:
                      02D5      ;       NONE
                      02D5      ;
                      02D5      ;
                      02D5      ; IMPLICIT OUTPUTS:
                      02D5      ;       NONE
                      02D5      ;
                      02D5      ;
                      02D5      ; COMPLETION CODES:
                      02D5      ;       NONE
                      02D5      ;
                      02D5      ; SIDE EFFECTS:
                      02D5      ;       OPCODE RESERVED TO DIGITAL error is possible, see ENVIRONMENT.
                      02D5      ;
                      02D5      ;--
                      02D5      LIB$$CVT_CVTHF_R1 ::
        61    60 F6FD 02D5              CVTHF   (R0), (R1)                      ;Convert H_FLOATING to FLOAT
                   05 02D9              RSB
```

```
                02DA          .SBTTL  LIB$$CVT_CVTHD_R1
                02DA
                02DA   ;++
                02DA   ; FUNCTIONAL DESCRIPTION:
                02DA   ;        Convert H_FLOATING to D_FLOATING
                02DA   ;
                02DA   ;
                02DA   ; CALLING SEQUENCE:
                02DA   ;        LIB$$CVT_CVTHD_R1 ( hfloat.rh.r, dfloat.wd.r )
                02DA   ;        This is a JSB entry point.
                02DA   ;
                02DA   ;
                02DA   ; FORMAL PARAMETERS:
                02DA   ;        R0 --> hfloat   R1 --> dfloat
                02DA   ;
                02DA   ;
                02DA   ; IMPLICIT INPUTS:
                02DA   ;        NONE
                02DA   ;
                02DA   ;
                02DA   ; IMPLICIT OUTPUTS:
                02DA   ;        NONE
                02DA   ;
                02DA   ;
                02DA   ; COMPLETION CODES:
                02DA   ;        NONE
                02DA   ;
                02DA   ; SIDE EFFECTS:
                02DA   ;        OPCODE RESERVED TO DIGITAL error is possible, see ENVIRONMENT.
                02DA   ;
                02DA   ;--
                02DA   LIB$$CVT_CVTHD_R1 ::
  61   60 F7FD  02DA          CVTHD   (R0), (R1)              ;Convert H_FLOATING to D_FLOATING
            05  02DE          RSB
```

```
                  02DF          .SBTTL  LIB$$CVT_CVTHG_R1
                  02DF
                  02DF     ;++
                  02DF     ; FUNCTIONAL DESCRIPTION:
                  02DF     ;        Convert H_FLOATING to G_FLOATING
                  02DF     ;
                  02DF     ;
                  02DF     ; CALLING SEQUENCE:
                  02DF     ;        LIB$$CVT_CVTHG_R1 ( hfloat.rh.r, gfloat.wg.r )
                  02DF     ;        This is a JSB entry point.
                  02DF     ;
                  02DF     ;
                  02DF     ; FORMAL PARAMETERS:
                  02DF     ;        R0 --> hfloat   R1 --> gfloat
                  02DF     ;
                  02DF     ;
                  02DF     ; IMPLICIT INPUTS:
                  02DF     ;        NONE
                  02DF     ;
                  02DF     ;
                  02DF     ; IMPLICIT OUTPUTS:
                  02DF     ;        NONE
                  02DF     ;
                  02DF     ;
                  02DF     ; COMPLETION CODES:
                  02DF     ;        NONE
                  02DF     ;
                  02DF     ; SIDE EFFECTS:
                  02DF     ;        OPCODE RESERVED TO DIGITAL error is possible, see ENVIRONMENT.
                  02DF     ;
                  02DF     ;--
                  02DF     LIB$$CVT_CVTHG_R1 ::
  61   60 76FD    02DF          CVTHG   (R0), (R1)              ;Convert H_FLOATING to G_FLOATING
             05   02E3          RSB
```

```
                  02E4        .SBTTL  LIB$$CVT_CVTGH_R1
                  02E4
                  02E4  ;++
                  02E4  ; FUNCTIONAL DESCRIPTION:
                  02E4  ;       Convert G_FLOATING to H_FLOATING
                  02E4  ;
                  02E4  ;
                  02E4  ; CALLING SEQUENCE:
                  02E4  ;       LIB$$CVT_CVTGH_R1 ( gfloat.rg.r, hfloat.wh.r )
                  02E4  ;       This is a JSB entry point.
                  02E4  ;
                  02E4  ;
                  02E4  ; FORMAL PARAMETERS:
                  02E4  ;       R0 --> gfloat   R1 --> hfloat
                  02E4  ;
                  02E4  ;
                  02E4  ; IMPLICIT INPUTS:
                  02E4  ;       NONE
                  02E4  ;
                  02E4  ;
                  02E4  ; IMPLICIT OUTPUTS:
                  02E4  ;       NONE
                  02E4  ;
                  02E4  ;
                  02E4  ; COMPLETION CODES:
                  02E4  ;       NONE
                  02E4  ;
                  02E4  ; SIDE EFFECTS:
                  02E4  ;       OPCODE RESERVED TO DIGITAL error is possible, see ENVIRONMENT.
                  02E4  ;
                  02E4  ;--
                  02E4  LIB$$CVT_CVTGH_R1 ::
    61   60 56FD  02E4        CVTGH   (R0), (R1)              ;Convert G_FLOATING TO H_FLOATING
               05 02E8        RSB
```

```
                            02E9          .SBTTL  LIB$$CVT_SCALE_OU_UP_BY_10_R1
                            02E9
                            02E9  ;++
                            02E9  ; FUNCTIONAL DESCRIPTION:
                            02E9  ;       Scale an unsigned OCTAWORD up by 10.
                            02E9  ;
                            02E9  ;
                            02E9  ; CALLING SEQUENCE:
                            02E9  ;       LIB$$CVT_SCALE_OU_UP_BY_10_R1 ( octa.mo.r )
                            02E9  ;       This is a JSB entry point.
                            02E9  ;
                            02E9  ;
                            02E9  ; FORMAL PARAMETERS:
                            02E9  ;       R0 --> octa
                            02E9  ;
                            02E9  ;
                            02E9  ; IMPLICIT INPUTS:
                            02E9  ;       NONE
                            02E9  ;
                            02E9  ;
                            02E9  ; IMPLICIT OUTPUTS:
                            02E9  ;       NONE
                            02E9  ;
                            02E9  ;
                            02E9  ; COMPLETION CODES:
                            02E9  ;       NONE
                            02E9  ;
                            02E9  ; SIDE EFFECTS:
                            02E9  ;
                            02E9  ;       NONE
                            02E9  ;--
                            02E9  LIB$$CVT_SCALE_OU_UP_BY_10_R1 ::
                  1F   BB   02E9          PUSHR   #^M<R0,R1,R2,R3,R4>
             54   04   D0   02EB          MOVL    #4, R4                  ; Do four longwords
                  53   D4   02EE          CLRL    R3                      ; Clear the 'carry'
51    53     60   0A   7A   02F0  10$:    EMUL    #10, (R0), R3, R1       ; Multiply low-order longwords
                  03   18   02F5          BGEQ    20$                     ; Make sure low-order longword
             52   0A   C0   02F7          ADDL2   #10, R2                 ;    is seen as unsigned
             80   51   D0   02FA  20$:    MOVL    R1, (R0)+               ; Store low longword
             53   52   D0   02FD          MOVL    R2, R3                  ; Save the 'carry'
             ED   54   F5   0300          SOBGTR  R4, 10$                 ; Continue looping
                  53   D5   0303          TSTL    R3                      ; Check for overflow
                  0E   13   0305          BEQL    30$                     ; Branch if no overflow
7E    0000047C    8F   D0   0307          MOVL    #SS$_INTOVF, -(SP)      ; Integer overflow
00000000'GF       01   FB   030E          CALLS   #1, G^LIB$SIGNAL        ; Signal the error
                  1F   BA   0315  30$:    POPR    #^M<R0,R1,R2,R3,R4>
                       05   0317          RSB                             ; Return
                            0318
```

```
                              0318          .SBTTL  LIB$$CVT_SCALE_OU_DOWN_BY_10_R1
                              0318
                              0318   ;++
                              0318   ; FUNCTIONAL DESCRIPTION:
                              0318   ;       Scales down an OCTAWORD by 10.
                              0318   ;
                              0318   ;
                              0318   ; CALLING SEQUENCE:
                              0318   ;       LIB$$CVT_SCALE_OU_DOWN_BY_10_R1 ( octa.mo.r )
                              0318   ;       This is a JSB entry point.
                              0318   ;
                              0318   ;
                              0318   ; FORMAL PARAMETERS:
                              0318   ;       R0 --> octa
                              0318   ;
                              0318   ;
                              0318   ; IMPLICIT INPUTS:
                              0318   ;       NONE
                              0318   ;
                              0318   ;
                              0318   ; IMPLICIT OUTPUTS:
                              0318   ;       NONE
                              0318   ;
                              0318   ;
                              0318   ; COMPLETION CODES:
                              0318   ;       NONE
                              0318   ;
                              0318   ; SIDE EFFECTS:
                              0318   ;       NONE
                              0318   ;
                              0318   ;--
                              0318   LIB$$CVT_SCALE_OU_DOWN_BY_10_R1::
                   1F   BB    0318          PUSHR   #^M<R0,R1,R2,R3,R4>
             54    03   D0    031A          MOVL    #3, R4                        ; Do four longwords
                   52   D4    031D          CLRL    R2                            ; Clear high longword of quad
          51     6044   D0    031F   10$:   MOVL    (R0)[R4], R1                  ; Grab low longword of quad
  52  6044   51    0A   7B    0323          EDIV    #10, R1, (R0)[R4], R2         ; Do a divide
             52    05   D1    0329   30$:   CMPL    #5, R2                        ; See if remainder too large
                   0D   1A    032C          BGTRU   50$                           ;    for the next EDIV
                   08   15    032E          BLEQ    40$                           ; Simply make remainder smaller
             52    0A   C0    0330          ADDL2   #10, R2                       ; Make remainder positive
                 6044   D7    0333          DECL    (R0)[R4]                      ; Make quotient smaller
                   F1   11    0336          BRB     30$                           ; Join the looping code
             52    0A   C2    0338   40$:   SUBL2   #10, R2                       ; Make remainder smaller
                E1 54   F4    033B   50$:   SOBGEQ  R4, 10$                       ; Continue looping
                   1F   BA    033E          POPR    #^M<R0,R1,R2,R3,R4>
                   05         0340          RSB                                   ; Return
                              0341
```

```
                    0341          .SBTTL  LIB$$CVT_MULD2_R1
                    0341
                    0341  ;++
                    0341  ; FUNCTIONAL DESCRIPTION:
                    0341  ;       multiply two DOUBLE floating values.
                    0341  ;
                    0341  ;
                    0341  ; CALLING SEQUENCE:
                    0341  ;       LIB$$CVT_MULD2_R1 ( mulr.rd.r, prod.md.r )
                    0341  ;       This is a JSB entry point.
                    0341  ;
                    0341  ;
                    0341  ; FORMAL PARAMETERS:
                    0341  ;       R0 --> mulr     R1 --> prod
                    0341  ;
                    0341  ;
                    0341  ; IMPLICIT INPUTS:
                    0341  ;       NONE
                    0341  ;
                    0341  ;
                    0341  ; IMPLICIT OUTPUTS:
                    0341  ;       NONE
                    0341  ;
                    0341  ;
                    0341  ; COMPLETION CODES:
                    0341  ;       NONE
                    0341  ;
                    0341  ; SIDE EFFECTS:
                    0341  ;       NONE
                    0341  ;
                    0341  ;--
                    0341  LIB$$CVT_MULD2_R1 ::
      61   60   64  0341          MULD2   (R0), (R1)              ;Multiply two D_FLOATING
                05  0344          RSB
```

```
                 0345          .SBTTL  LIB$$CVT_DIVD2_R1
                 0345
                 0345   ;++
                 0345   ; FUNCTIONAL DESCRIPTION:
                 0345   ;       Divide two DOUBLE floating values.
                 0345   ;
                 0345   ;
                 0345   ; CALLING SEQUENCE:
                 0345   ;       LIB$$CVT_DIVD2_R1 ( divr.rd.r, quo.md.r )
                 0345   ;       This is a JSB entry point.
                 0345   ;
                 0345   ;
                 0345   ; FORMAL PARAMETERS:
                 0345   ;       R0 --> divr     R1 --> quo
                 0345   ;
                 0345   ;
                 0345   ; IMPLICIT INPUTS:
                 0345   ;       NONE
                 0345   ;
                 0345   ;
                 0345   ; IMPLICIT OUTPUTS:
                 0345   ;       NONE
                 0345   ;
                 0345   ;
                 0345   ; COMPLETION CODES:
                 0345   ;       NONE
                 0345   ;
                 0345   ; SIDE EFFECTS:
                 0345   ;       NONE
                 0345   ;
                 0345   ;--
                 0345   LIB$$CVT_DIVD2_R1 ::
   61   60   66  0345          DIVD2   (R0), (R1)              ;Divide two D_FLOATING
             05  0348          RSB
```

```
               0349         .SBTTL  LIB$$CVT_MULH2_R1
               0349
               0349  ;++
               0349  ; FUNCTIONAL DESCRIPTION:
               0349         multiply two H_FLOATING floating values.
               0349  ;
               0349  ;
               0349  ; CALLING SEQUENCE:
               0349  ;      LIB$$CVT_MULH2_R1 ( mulr.rh.r, prod.mh.r )
               0349  ;      This is a JSB entry point.
               0349  ;
               0349  ;
               0349  ; FORMAL PARAMETERS:
               0349  ;      R0 --> mulr     R1 --> prod
               0349  ;
               0349  ;
               0349  ; IMPLICIT INPUTS:
               0349         NONE
               0349  ;
               0349  ;
               0349  ; IMPLICIT OUTPUTS:
               0349         NONE
               0349  ;
               0349  ;
               0349  ; COMPLETION CODES:
               0349         NONE
               0349  ;
               0349  ; SIDE EFFECTS:
               0349  ;
               0349  ;      OPCODE RESERVED TO DIGITAL error is possible, see ENVIRONMENT.
               0349  ;
               0349  ;--
               0349  LIB$$CVT_MULH2_R1 ::
61   60 64FD   0349         MULH2   (R0), (R1)               ;Multiply two H_FLOATING
           05  034D         RSB
```

```
                 034E          .SBTTL  LIB$$CVT_DIVH2_R1
                 034E
                 034E  ;++
                 034E  ; FUNCTIONAL DESCRIPTION:
                 034E  ;       Divide two H_FLOATING floating values.
                 034E  ;
                 034E  ;
                 034E  ; CALLING SEQUENCE:
                 034E  ;       LIB$$CVT_DIVH2_R1 ( divr.rh.r, quo.mh.r )
                 034E  ;       This is a JSB entry point.
                 034E  ;
                 034E  ;
                 034E  ; FORMAL PARAMETERS:
                 034E  ;       R0 --> divr      R1 --> quo
                 034E  ;
                 034E  ;
                 034E  ; IMPLICIT INPUTS:
                 034E  ;       NONE
                 034E  ;
                 034E  ;
                 034E  ; IMPLICIT OUTPUTS:
                 034E  ;       NONE
                 034E  ;
                 034E  ;
                 034E  ; COMPLETION CODES:
                 034E  ;       NONE
                 034E  ;
                 034E  ; SIDE EFFECTS:
                 034E  ;
                 034E  ;       OPCODE RESERVED TO DIGITAL error is possible, see ENVIRONMENT.
                 034E  ;
                 034E  ;--
                 034E  LIB$$CVT_DIVH2_R1 ::
61    60 66FD    034E          DIVH2   (R0), (R1)               ;Divide two H_FLOATING
            05   0352          RSB
```

```
                                 0353            .SBTTL  LIB$$CVT_ASHP_R1
                                 0353
                                 0353     ;++
                                 0353     ; FUNCTIONAL DESCRIPTION:
                                 0353     ;       Scale a packed decimal number.
                                 0353     ;
                                 0353     ;
                                 0353     ; CALLING SEQUENCE:
                                 0353     ;       LIB$$CVT_ASHP_R1 (cnt.rw.r, srclen.rw.r, srcaddr.ra.r,
                                 0353     ;                         round.rb.r, dstlen.rw.r, dstaddr.ra.r )
                                 0353     ;       This is a JSB entry point.
                                 0353     ;
                                 0353     ;
                                 0353     ; FORMAL PARAMETERS:
                                 0353     ;       R0 --> cnt      R1 = srclen       R2 --> srcaddr  R3 = round
                                 0353     ;       R4 --> dstlen   R5 = dstaddr
                                 0353     ;
                                 0353     ;
                                 0353     ; IMPLICIT INPUTS:
                                 0353     ;       NONE
                                 0353     ;
                                 0353     ;
                                 0353     ; IMPLICIT OUTPUTS:
                                 0353     ;       NONE
                                 0353     ;
                                 0353     ;
                                 0353     ; COMPLETION CODES:
                                 0353     ;       NONE
                                 0353     ;
                                 0353     ; SIDE EFFECTS:
                                 0353     ;       NONE
                                 0353     ;
                                 0353     ;--
                                 0353     LIB$$CVT_ASHP_R1 ::
                         3F  BB  0353            PUSHR   #^M<R0,R1,R2,R3,R4,R5>
65  64  63  62  61       60  F8  0355            ASHP    (R0), (R1), (R2), (R3), (R4), (R5)
                         3F  BA  035C            POPR    #^M<R0,R1,R2,R3,R4,R5>
                             05  035E            RSB
```

```
                         035F          .SBTTL  LIB$$CVT_CMPH_R1
                         035F
                         035F  ;++
                         035F  ; FUNCTIONAL DESCRIPTION:
                         035F  ;       Compare two H floating values
                         035F  ;
                         035F  ;
                         035F  ; CALLING SEQUENCE:
                         035F  ;       STATUS = LIB$$CVT_CMPH_R1 ( src1.rh.r, src2.rh.r )
                         035F  ;       This is a JSB entry point.
                         035F  ;
                         035F  ;
                         035F  ; FORMAL PARAMETERS:
                         035F  ;       R0 --> src1       R1 --> src2
                         035F  ;
                         035F  ;
                         035F  ; IMPLICIT INPUTS:
                         035F  ;       NONE
                         035F  ;
                         035F  ;
                         035F  ; IMPLICIT OUTPUTS:
                         035F  ;       NONE
                         035F  ;
                         035F  ;
                         035F  ; COMPLETION CODES:
                         035F  ;       R0 = -1 if src1 < src2
                         035F  ;       R0 = 0  if src1 = src2
                         035F  ;       R0 = 1  if src1 > src2
                         035F  ;
                         035F  ; SIDE EFFECTS:
                         035F  ;       OPCODE RESERVED TO DIGITAL error is possible, see ENVIRONMENT.
                         035F  ;
                         035F  ;--
                         035F  LIB$$CVT_CMPH_R1 ::
        61    60 71FD    035F          CMPH    (R0), (R1)
              06   13    0363          BEQL    10$
              07   19    0365          BLSS    20$
        50    01   CE    0367          MNEGL   #1, R0                          ;src1 > src2
                   05    036A          RSB                                     ;Finished
                         036B  10$:
              50   D4    036B          CLRL    R0                              ;They are equal
                   05    036D          RSB                                     ;Finished
                         036E  20$:
  50  FFFFFFFF 8F  D0    036E          MOVL    #-1, R0                         ;src1 < src2
                   05    0375          RSB                                     ;Finished
                         0376
```

```
                0376  1272 .END
```

```
D32                                00000062 R     01
DDUMMY                             0000004A R     01
D_OFF                            = 00000007
F32                                00000019 R     01
FDUMMY                             0000000D R     01
G32                                000000B3 R     01
GDUMMY                             0000009B R     01
LIB$$CVT_ASHP_R1                   00000353 RG    01
LIB$$CVT_CMPH_R1                   0000035F RG    01
LIB$$CVT_CVTDH_R1                  000001D8 RG    01
LIB$$CVT_CVTGH_R1                  000002E4 RG    01
LIB$$CVT_CVTHD_R1                  000002DA RG    01
LIB$$CVT_CVTHF_R1                  000002D5 RG    01
LIB$$CVT_CVTHG_R1                  000002DF RG    01
LIB$$CVT_CVTLB_R1                  00000000 RG    01
LIB$$CVT_CVTLH_R1                  00000008 RG    01
LIB$$CVT_CVTLW_R1                  00000004 RG    01
LIB$$CVT_CVTRDH_R1                 000001A8 RG    01
LIB$$CVT_CVTRHL_R1                 000001DD RG    01
LIB$$CVT_CVTRHO_R1                 0000027F RG    01
LIB$$CVT_CVTRHQ_R1                 000001E2 RG    01
LIB$$CVT_CVTROUD_R1                00000072 RG    01
LIB$$CVT_CVTROUF_R1                00000021 RG    01
LIB$$CVT_CVTROUG_R1                000000C3 RG    01
LIB$$CVT_CVTROUH_R1                000000F0 RG    01
LIB$$CVT_DIVD2_R1                  00000345 RG    01
LIB$$CVT_DIVH2_R1                  0000034E RG    01
LIB$$CVT_MULD2_R1                  00000341 RG    01
LIB$$CVT_MULH2_R1                  00000349 RG    01
LIB$$CVT_SCALE_OU_DOWN_BY_10_R1    00000318 RG    01
LIB$$CVT_SCALE_OU_UP_BY_10_R1      000002E9 RG    01
LIB$SIGNAL                         ********    X  01
O_LEN                            = 00000004
SHIFT_OU_LEFT                      0000018C R     01
SS$_INTOVF                       = 0000047C

                                   +----------------+
                                   ! Psect synopsis !
                                   +----------------+

PSECT name                         Allocation             PSECT No.  Attributes
----------                         ----------             ---------  ----------

.  ABS  .                          00000000  (     0.)    00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD  NOWRT NOVEC BYTE
_LIB$CODE                          00000376  (   886.)    01 (  1.)    PIC   USR   CON   REL   LCL   SHR   EXE   RD  NOWRT NOVEC LONG
$ABS$                              00000000  (     0.)    02 (  2.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD    WRT NOVEC BYTE

                                +------------------------+
                                ! Performance indicators !
                                +------------------------+

Phase                    Page faults    CPU Time       Elapsed Time
-----                    -----------    --------       ------------
Initialization                    29    00:00:00.04    00:00:02.60
Command processing               121    00:00:00.33    00:00:05.82
Pass 1                           229    00:00:03.60    00:00:25.63
Symbol table sort                  0    00:00:00.47    00:00:05.50
Pass 2                           209    00:00:01.68    00:00:07.80
```

```
Symbol table output            5     00:00:00.04    00:00:00.04
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         598     00:00:06.18    00:00:47.40

The working set limit was 1200 pages.
34798 bytes (68 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 441 non-local and 44 local symbols.
1272 source lines were read in Pass 1, producing 12 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

469 GETS were required to define 4 macros.

There were no errors, warnings or information messages.

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

0204 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

LIBCVTDX
LIS

LIBCVTMAC
LIS

LIBCRETAB
LIS

LIBCVTDF
LIS

LIBCUSTOM
LIS

LIBCREDIR
LIS

LIBCVTATB
LIS

LIBCOMMON
LIS

LIBCRC
LIS

0205 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

LIBCVTTPU
LIS

LIBCVTUO
LIS

LIBDECOVE
LIS

LIBDAY
LIS

LIBCVTOU
LIS

LIBCVTPTU
LIS

LIBCVTTPO
LIS

LIBDECODF
LIS

LIBDOCOM
LIS

LIBCVTTPZ
LIS

LIBDATETI
LIS

LIBDELETE
LIS

LIBCVTPTO
LIS

LIBDAYWK
LIS

LIBCVTPTZ
LIS