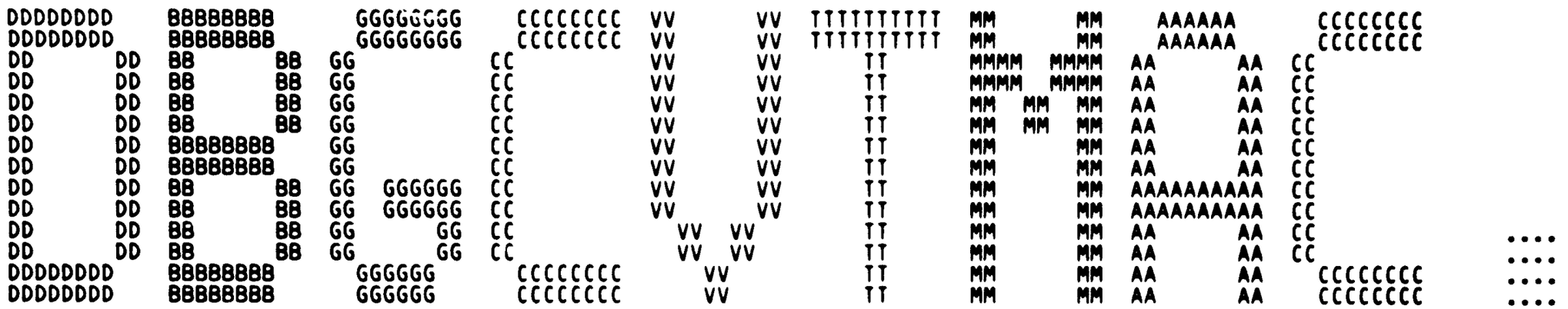
```
DDDDDDDDDDD     EEEEEEEEEEEEEEE  BBBBBBBBBBB      UUU         UUU     GGGGGGGGGGG
DDDDDDDDDDDD    EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUU         UUU     GGGGGGGGGGGG
DDDDDDDDDDDD    EEEEEEEEEEEEEEE  BBBBBBBBBBBB     UUU         UUU     GGGGGGGGGGGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG
DDD         DDD EEEEEEEEEEE      BBBBBBBBBBBB     UUU         UUU GGG
DDD         DDD EEEEEEEEEEE      BBBBBBBBBBBB     UUU         UUU GGG
DDD         DDD EEEEEEEEEEE      BBBBBBBBBBBB     UUU         UUU GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG   GGGGGGGGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG   GGGGGGGGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG   GGGGGGGGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG         GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG         GGG
DDD         DDD EEE              BBB         BBB  UUU         UUU GGG         GGG
DDDDDDDDDDDD    EEEEEEEEEEEEEEE  BBBBBBBBBBBB      UUUUUUUUUUUUUUUUU     GGGGGGGGG
DDDDDDDDDDDD    EEEEEEEEEEEEEEE  BBBBBBBBBBBB      UUUUUUUUUUUUUUUUU     GGGGGGGGG
DDDDDDDDDDDD    EEEEEEEEEEEEEEE  BBBBBBBBBBBB      UUUUUUUUUUUUUUUUU     GGGGGGGGG
```


DDDDDDDD BBBBBBBB GGGGGGGG CCCCCCCC VV VV TTTTTTTTTT MM MM AAAAAA CCCCCCCC
DDDDDDDD BBBBBBBB GGGGGGGG CCCCCCCC VV VV TTTTTTTTTT MM MM AAAAAA CCCCCCCC
DD DD BB BB GG CC VV VV TT MMMM MMMM AA AA CC
DD DD BB BB GG CC VV VV TT MMMM MMMM AA AA CC
DD DD BB BB GG CC VV VV TT MM MM MM AA AA CC
DD DD BB BB GG CC VV VV TT MM MM MM AA AA CC
DD DD BBBBBBBB GG CC VV VV TT MM MM AA AA CC
DD DD BBBBBBBB GG CC VV VV TT MM MM AA AA CC
DD DD BB BB GG GGGGGG CC VV VV TT MM MM AAAAAAAAAA CC
DD DD BB BB GG GGGGGG CC VV VV TT MM MM AAAAAAAAAA CC
DD DD BB BB GG GG CC VV VV TT MM MM AA AA CC
DD DD BB BB GG GG CC VV VV TT MM MM AA AA CC
DDDDDDDD BBBBBBBB GGGGGG CCCCCCCC VV TT MM MM AA AA CCCCCCCC
DDDDDDDD BBBBBBBB GGGGGG CCCCCCCC VV TT MM MM AA AA CCCCCCCC

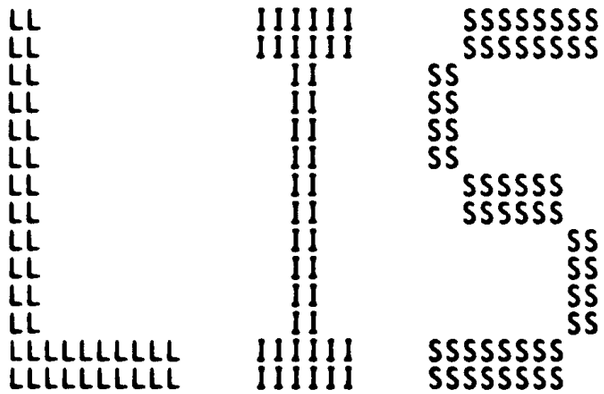

LL IIIIII SSSSSSSS
LL IIIIII SSSSSSSS
LL II SS
LL II SS
LL II SS
LL II SS
LL II SSSSSS
LL II SSSSSS
LL II SS
LL II SS
LL II SS
LLLLLLLLLL IIIIII SSSSSSSS
LLLLLLLLLL IIIIII SSSSSSSS

Table of contents

```
0000     1          .TITLE  LIB$CVTMAC      MACRO-32 support for LIB$CVT_DX_DX
0000     2          .IDENT  /V04-000/       ; File: LIBCVTMAC.MAR  EDIT: FMT002
0000     3
0000     4  ;
0000     5  ;*****************************************************************************
0000     6  ;*                                                                           *
0000     7  ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     8  ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     9  ;*  ALL RIGHTS RESERVED.                                                     *
0000    10  ;*                                                                           *
0000    11  ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED     *
0000    12  ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
0000    13  ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER     *
0000    14  ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY     *
0000    15  ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY     *
0000    16  ;*  TRANSFERRED.                                                             *
0000    17  ;*                                                                           *
0000    18  ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE     *
0000    19  ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
0000    20  ;*  CORPORATION.                                                             *
0000    21  ;*                                                                           *
0000    22  ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS     *
0000    23  ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    24  ;*                                                                           *
0000    25  ;*                                                                           *
0000    26  ;*****************************************************************************
0000    27  ;
0000    28  ;++
0000    29  ; FACILITY: Run-time library.
0000    30  ;
0000    31  ; ABSTRACT:
0000    32  ;       This module contains the MACRO-32 support routines for LIB$CVT_DX_DX
0000    33  ;       routine.  All of the entry points are JSB entry points.
0000    34  ;
0000    35  ; ENVIRONMENT:
0000    36  ;       Some entry points require the Caller to have LIB$EMULATE or the
0000    37  ;       machine have the full instruction set of VAX-11 architecture.
0000    38  ;       Note that this module is for specific support of LIB$CVT_DX_DX routine,
0000    39  ;       hence it has no  RTL "globally defined" entry point, therefore this
0000    40  ;       module assumes that the caller has IV, FU, DV bits in PSW turned on.
0000    41  ;
0000    42  ; AUTHOR:
0000    43  ;       AUTHORS :  FAROKH MORSHED, PETER GILBERT  7-FEB-81, VERSION 1, VMS V3.0.
0000    44  ;
0000    45  ; MODIFIED BY:
0000    46  ;
0000    47  ; VERSION: 1
0000    48  ; 1001  Original.
0000    49  ; 1002  Make references to LIB$SIGNAL General mode addressing.  FM 30-OCT-81.
0000    50  ;       P.S.    30-Sep-83       fixed a bug reported from run time library
0000    51  ;                               SPR.  Convert quadword to floating data type,
0000    52  ;                               if the low longword in quadword has 0 then
0000    53  ;                               the floating value is number * 2**32.
0000    54  ;--
0000    55  ;
```

```
                     0000     57          .SBTTL  DECLARATIONS
                     0000     58
                     0000     59 ;
                     0000     60 ; PSECT DECLARATIONS:
                     0000     61 ;
                     0000     62          .PSECT  _LIB$CODE       PIC, SHR, LONG, EXE, NOWRT
                     0000     63 ;
                     0000     64 ; EXTERNAL ROUTINES
                     0000     65 ;
                     0000     66          .EXTERNAL       LIB$SIGNAL
                     0000     67 ;
                     0000     68 ;        MACRO
                     0000     69 ;
                     0000     70          $SSDEF
                     0000     71
```

```
                                        .SBTTL  DBG$CVT_CVTLB_R1

                                ;++
                                ; FUNCTIONAL DESCRIPTION:
                                ;       Convert LONGWORD to BY^E
                                ;
                                ; CALLING SEQUENCE:
                                ;       DBG$CVT_CVTLB_R1 ( long.rl.r, byte.wb.r )
                                ;       This is a JSB entry point.
                                ;
                                ; FORMAL PARAMETERS:
                                ;       R0 --> long     R1 --> byte
                                ;
                                ; IMPLICIT INPUTS:
                                ;       NONE
                                ;
                                ; IMPLICIT OUTPUTS:
                                ;       NONE
                                ;
                                ; COMPLETION CODES:
                                ;       NONE
                                ;
                                ; SIDE EFFECTS:
                                ;
                                ;--

                                DBG$CVT_CVTLB_R1 ::
61    60    F6    0000    106           CVTLB   (R0), (R1)              ;Convert LONGWORD to BYTE
                05  0003    107           RSB
```

```
                     0004   109          .SBTTL  DBG$CVT_CVTLW_R1
                     0004   110
                     0004   111 ;++
                     0004   112 ; FUNCTIONAL DESCRIPTION:
                     0004   113 ;       Convert LONGWORD to WORD.
                     0004   114 ;
                     0004   115 ;
                     0004   116 ; CALLING SEQUENCE:
                     0004   117 ;       DBG$CVT_CVTLW_R1 ( long.rl.r, word.ww.r )
                     0004   118 ;       This is a JSB entry point.
                     0004   119 ;
                     0004   120 ;
                     0004   121 ; FORMAL PARAMETERS:
                     0004   122 ;       R0 --> long     R1 --> word
                     0004   123 ;
                     0004   124 ;
                     0004   125 ; IMPLICIT INPUTS:
                     0004   126 ;       NONE
                     0004   127 ;
                     0004   128 ;
                     0004   129 ; IMPLICIT OUTPUTS:
                     0004   130 ;       NONE
                     0004   131 ;
                     0004   132 ;
                     0004   133 ; COMPLETION CODES:
                     0004   134 ;       NONE
                     0004   135 ;
                     0004   136 ; SIDE EFFECTS:
                     0004   137 ;
                     0004   138 ;
                     0004   139 ;--
                     0004   140
                     0004   141 DBG$CVT_CVTLW_R1 ::
  61    60    F7     0004   142          CVTLW   (R0), (R1)              ;Convert LONGWORD to WORD
                05   0007   143          RSB
```

```
                     0008   145          .SBTTL  DBG$CVT_CVTLH_R1
                     0008   146
                     0008   147 ;++
                     0008   148 ; FUNCTIONAL DESCRIPTION:
                     0008   149 ;       Convert LONGWORD to H_FLOATING.
                     0008   150 ;
                     0008   151 ;
                     0008   152 ; CALLING SEQUENCE:
                     0008   153 ;       DBG$CVT_CVTLH_R1 ( long.rl.r, hfloat.wh.r )
                     0008   154 ;       This is a JSB entry point.
                     0008   155 ;
                     0008   156 ;
                     0008   157 ; FORMAL PARAMETERS:
                     0008   158 ;       R0 --> long     R1 --> hfloat
                     0008   159 ;
                     0008   160 ;
                     0008   161 ; IMPLICIT INPUTS:
                     0008   162 ;       NONE
                     0008   163 ;
                     0008   164 ;
                     0008   165 ; IMPLICIT OUTPUTS:
                     0008   166 ;       NONE
                     0008   167 ;
                     0008   168 ;
                     0008   169 ; COMPLETION CODES:
                     0008   170 ;       NONE
                     0008   171 ;
                     0008   172 ; SIDE EFFECTS:
                     0008   173 ;       OPCODE RESERVED TO DIGITAL error is possible, see ENVIRONMENT.
                     0008   174 ;
                     0008   175 ;--
                     0008   176
                     0008   177 DBG$CVT_CVTLH_R1 ::
61    60 6EFD        0008   178          CVTLH   (R0), (R1)              ;Convert LONGWORD to H_FLOATING
         05          000C   179          RSB
                     000D   180
```

```
                                000D   182          .SBTTL  DBG$CVT_CVTROUF_R1
                                000D   183
                                000D   184 ;++
                                000D   185 ; FUNCTIONAL DESCRIPTION:
                                000D   186 ;       Convert unsigned OCTAWORD to FLOAT.
                                000D   187 ;
                                000D   188 ;
                                000D   189 ; CALLING SEQUENCE:
                                000D   190 ;       DBG$CVT_CVTROUF_R1 ( octa.ro.r, float.wf.r )
                                000D   191 ;       This is a JSB entry point.
                                000D   192 ;
                                000D   193 ;
                                000D   194 ; FORMAL PARAMETERS:
                                000D   195 ;       R0 --> octa     R1 --> float
                                000D   196 ;
                                000D   197 ;
                                000D   198 ; IMPLICIT INPUTS:
                                000D   199 ;       NONE
                                000D   200 ;
                                000D   201 ;
                                000D   202 ; IMPLICIT OUTPUTS:
                                000D   203 ;       NONE
                                000D   204 ;
                                000D   205 ;
                                000D   206 ; COMPLETION CODES:
                                000D   207 ;       NONE
                                000D   208 ;
                                000D   209 ; SIDE EFFECTS:
                                000D   210 ;       OPCODE RESERVED TO DIGITAL error is possible, see ENVIRONMENT.
                                000D   211 ;
                                000D   212 ;--
                                000D   213
                                000D   214 ;
                                000D   215 ;         DEFINE SOME CONSTANTS
                                000D   216 ;
                    00000011    000D   217 FDUMMY:  .BLKL           1
                    00007080    0011   218          .F_FLOATING    792281625142643375935439503336   ;2**96
                    00006080    0015   219          .F_FLOATING    18446744073709551616             ;2**64
                    00005080    0019   220 F32:     .F_FLOATING    4294967296                       ;2**32
                    00004080    001D   221          .F_FLOATING    1                                ;
                                0021   222 ;
                                0021   223 ;         FIND THE FLOATING ATOMIC DATA TYPE BY MULTIPLYING EACH LONGWORD
                                0021   224 ;         OF OCTAWORD BY AN APPROPRIATE CONSTANT.
                                0021   225 ;
                                0021   226 DBG$CVT_CVTROUF_R1 ::
                      0F    BB  0021   227          PUSHR   #^M<R0,R1,R2,R3>
                      61    D4  0023   228          CLRF    (R1)                    ;Initialize destination.
                52    04    D0  0025   229          MOVL    #4, R2                  ;Set up the loop counter for 4 times
                53    60    4E  0028   230 10$:     CVTLF   (R0), R3                ;Convert the next LONGWORD to floating
                      06    14  002B   231          BGTR    13$                     ;This longword is positive
                      0C    13  002D   232          BEQL    15$
             53    E7 AF    40  002F   233          ADDF2   F32, R3                 ;Negative, so add the difference
          53    D6 AF42    44  0033   234 13$:     MULF2   FDUMMY[R2], R3          ;Multiply the result by the constant
                61    53    40  0038   235          ADDF2   R3, (R1)                ;Add in the result to destination
                      52    97  003B   236 15$:     DECB    R2                      ;One less time to go around in the loop
                      08    13  003D   237          BEQL    20$                     ;Finished if counter is zero
                      80    D5  003F   238          TSTL    (R0)+                   ;Next longword of OCTA
```

```
60   D5  0041   239          TSTL    (R0)                    ;Is it zero ?
F6   13  0043   240          BEQL    15$                     ;Yes, so ignore it
E1   11  0045   241          BRB     10$                     ;Loop to chomp some more on the OCTA
         0047   242 20$:
0F   BA  0047   243          POPR    #^M<R0,R1,R2,R3>
     05  0049   244          RSB
```

```
                                    004A   246          .SBTTL  DBG$CVT_CVTROUD_R1
                                    004A   247
                                    004A   248 ;++
                                    004A   249 ; FUNCTIONAL DESCRIPTION:
                                    004A   250 ;       Convert unsigned OCTAWORD to D_FLOATING.
                                    004A   251 ;
                                    004A   252 ;
                                    004A   253 ; CALLING SEQUENCE:
                                    004A   254 ;       DBG$CVT_CVTROUD_R1 ( octa.ro.r, dfloat.wd.r )
                                    004A   255 ;       This is a JSB entry point.
                                    004A   256 ;
                                    004A   257 ;
                                    004A   258 ; FORMAL PARAMETERS:
                                    004A   259 ;       R0 --> octa     R1 --> dfloat
                                    004A   260 ;
                                    004A   261 ;
                                    004A   262 ; IMPLICIT INPUTS:
                                    004A   263 ;       NONE
                                    004A   264 ;
                                    004A   265 ;
                                    004A   266 ; IMPLICIT OUTPUTS:
                                    004A   267 ;       NONE
                                    004A   268 ;
                                    004A   269 ;
                                    004A   270 ; COMPLETION CODES:
                                    004A   271 ;       NONE
                                    004A   272 ;
                                    004A   273 ; SIDE EFFECTS:
                                    004A   274 ;       OPCODE RESERVED TO DIGITAL error is possible, see ENVIRONMENT.
                                    004A   275 ;
                                    004A   276 ;--
                                    004A   277
                                    004A   278 ;
                                    004A   279 ;         DEFINE SOME CONSTANTS
                                    004A   280 ;
            00000052                004A   281 DDUMMY:  .BLKL           2
00000000 00007080                   0052   282          .D_FLOATING     79228162514264337593543950336   ;2**96
00000000 00006080                   005A   283          .D_FLOATING     18446744073709551616            ;2**64
00000000 00005080                   0062   284 D32:     .D_FLOATING     4294967296                      ;2**32
00000000 00004080                   006A   285          .D_FLOATING     1                               ;
                                    0072   286 ;
                                    0072   287 ;         FIND THE FLOATING ATOMIC DATA TYPE BY MULTIPLYING EACH LONGWORD
                                    0072   288 ;         OF OCTAWORD BY AN APPROPRIATE CONSTANT.
                                    0072   289 ;
                                    0072   290 DBG$CVT_CVTROUD_R1::
                 1F    BB           0072   291          PUSHR   #^M<R0,R1,R2,R3,R4>
                 61    7C           0074   292          CLRD    (R1)                    ;Initialize destination
           52    04    D0           0076   293          MOVL    #4, R2                  ;Set up the loop counter for 4 times
           53    60    6E           0079   294 10$:     CVTLD   (R0), R3                ;Convert the next LONGWORD to floating
                 06    14           007C   295          BGTR    13$                     ;This longword is positive
                 0C    13           007E   296          BEQL    15$
     53    DF  AF    60           0080   297          ADDD2   D32, R3                 ;Negative, so add the difference
53         C2  AF42    64           0084   298 13$:     MULD2   DDUMMY[R2], R3          ;Multiply the result by the constant
           61    53    60           0089   299          ADDD2   R3, (R1)                ;Add in the result to destination
                 52    97           008C   300 15$:     DECB    R2                      ;One less time to go around in the loop
                 08    13           008E   301          BEQL    20$                     ;Finished if counter is zero
                 80    D5           0090   302          TSTL    (R0)+                   ;Next longword of OCTA
```

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

```
                     009B   310          .SBTTL  DBG$CVT_CVTROUG_R1
                     009B   311
                     009B   312 ;++
                     009B   313 ; FUNCTIONAL DESCRIPTION:
                     009B   314 ;       Convert unsigned OCTAWORD to G_FLOATING
                     009B   315 ;
                     009B   316 ;
                     009B   317 ; CALLING SEQUENCE:
                     009B   318 ;       DBG$CVT_CVTROUG_R1 ( octa.ro.r, gfloat.wg.r )
                     009B   319 ;       This is a JSB entry point.
                     009B   320 ;
                     009B   321 ;
                     009B   322 ; FORMAL PARAMETERS:
                     009B   323 ;       R0 --> octa     R1 --> gfloat
                     009B   324 ;
                     009B   325 ;
                     009B   326 ; IMPLICIT INPUTS:
                     009B   327 ;       NONE
                     009B   328 ;
                     009B   329 ;
                     009B   330 ; IMPLICIT OUTPUTS:
                     009B   331 ;       NONE
                     009B   332 ;
                     009B   333 ;
                     009B   334 ; COMPLETION CODES:
                     009B   335 ;       NONE
                     009B   336 ;
                     009B   337 ; SIDE EFFECTS:
                     009B   338 ;       OPCODE RESERVED TO DIGITAL error is possible, see ENVIRONMENT.
                     009B   339 ;
                     009B   340 ;--
                     009B   341
                     009B   342 ;
                     009B   343 ;         DEFINE SOME CONSTANTS
                     009B   344 ;
             000000A3 009B   345 GDUMMY: .BLKL            2
00000000 00004610    00A3   346         .G_FLOATING     79228162514264337593543950336;2**96
00000000 00004410    00AB   347         .G_FLOATING     18446744073709551616;2**64
00000000 00004210    00B3   348 G32:     .G_FLOATING     4294967296;2**32
00000000 00004010    00BB   349         .G_FLOATING     1;
                     00C3   350 ;
                     00C3   351 ;         FIND THE FLOATING ATOMIC DATA TYPE BY MULTIPLYING EACH LONGWORD
                     00C3   352 ;         OF OCTAWORD BY AN APPROPRIATE CONSTANT.
                     00C3   353 ;
                     00C3   354 DBG$CVT_CVTROUG_R1 ::
          1F    BB   00C3   355         PUSHR   #^M<R0,R1,R2,R3,R4>
          61    7C   00C5   356         CLRG    (R1)                    ;Initialize destination
     52   04    D0   00C7   357         MOVL    #4, R2                  ;Set up the loop counter for 4 times
     53   60 4EFD    00CA   358 10$:     CVTLG   (R0), R3                ;Convert the next LONGWORD to floating
          07    14   00CE   359         BGTR    13$                     ;This longword is positive
          0F    13   00D0   360         BEQL    15$
  53    DD AF 40FD    00D2   361         ADDG2   G32, R3                 ;Negative, so add the difference
53    BF AF42 44FD    00D7   362 13$:     MULG2   GDUMMY[R2], R3          ;Multiply the result by the constant
     61   53 40FD    00DD   363         ADDG2   R3, (R1)                ;Add in the result to destination
          52    97   00E1   364 15$:     DECB    R2                      ;One less time to go around in the loop
          08    13   00E3   365         BEQL    20$                     ;Finished if counter is zero
          80    D5   00E5   366         TSTL    (R0)+                   ;Next longword of OCTA
```

```
                00E7    367          TSTL    (R0)                    ;Is it zero ?
F6   13         00E9    368          BEQL    15$                     ;Yes, so ignore it
DD   11         00EB    369          BRB     10$                     ;Loop to chomp some more on the OCTA
                00ED    370 20$:
1F   BA         00ED    371          POPR    #^M<R0,R1,R2,R3,R4>
     05         00EF    372          RSB
```

```
                                                00F0   374          .SBTTL  DBG$CVT_CVTROUH_R1
                                                00F0   375
                                                00F0   376 ;++
                                                00F0   377 ; FUNCTIONAL DESCRIPTION:
                                                00F0   378 ;       Convert unsigned OCTAWORD to H_FLOATING rounded.
                                                00F0   379 ;
                                                00F0   380 ;
                                                00F0   381 ; CALLING SEQUENCE:
                                                00F0   382 ;       DBG$CVT_CVTROUH_R1 ( octa.ro.r, hfloat.wh.r )
                                                00F0   383 ;       This is a JSB entry point.
                                                00F0   384 ;
                                                00F0   385 ;
                                                00F0   386 ; FORMAL PARAMETERS:
                                                00F0   387 ;       R0 --> octa     R1 --> hfloat
                                                00F0   388 ;
                                                00F0   389 ;
                                                00F0   390 ; IMPLICIT INPUTS:
                                                00F0   391 ;       NONE
                                                00F0   392 ;
                                                00F0   393 ;
                                                00F0   394 ; IMPLICIT OUTPUTS:
                                                00F0   395 ;       NONE
                                                00F0   396 ;
                                                00F0   397 ;
                                                00F0   398 ; COMPLETION CODES:
                                                00F0   399 ;       NONE
                                                00F0   400 ;
                                                00F0   401 ; SIDE EFFECTS:
                                                00F0   402 ;
                                                00F0   403 ;--
                                                00F0   404
                                                00F0   405 DBG$CVT_CVTROUH_R1 ::
                                          0F    BB    00F0   406          PUSHR   #^M<R0,R1,R2,R3>
                                          60    D5    00F2   407          TSTL    (R0)                    ;If OU is zero, return zero.
                                          1A    12    00F4   408          BNEQ    5$                      ;....
                                       04 A0    D5    00F6   409          TSTL    4(R0)                   ;....
                                          15    12    00F9   410          BNEQ    5$                      ;....
                                       08 A0    D5    00FB   411          TSTL    8(R0)                   ;....
                                          10    12    00FE   412          BNEQ    5$                      ;....
                                       0C A0    D5    0100   413          TSTL    12(R0)                  ;....
                                          0B    12    0103   414          BNEQ    5$                      ;....
                                          81    D4    0105   415          CLRL    (R1)+                   ;Return 0 in H.
                                          81    D4    0107   416          CLRL    (R1)+                   ;....
                                          81    D4    0109   417          CLRL    (R1)+                   ;....
                                          61    D4    010B   418          CLRL    (R1)                    ;....
                                        0079    31    010D   419          BRW     50$                     ;Go to RSB back to caller.
                                                      0110   420 5$:
                                       5E     12    C2 0110   421          SUBL2   #18, SP                 ;Make room for nomalized OU.
                                                      0113   422 ;
                                                      0113   423 ; Move source to top of stack to normalize it
                                                      0113   424 ;
                                       6E     60    7D 0113   425          MOVQ    (R0), (SP)              ;Put OU in here to be
                                    08 AF   08 A0    7D 0116   426          MOVQ    8(R0), 8(SP)            ;       worked on.
                                          10 AE    B4 011B   427          CLRW    16(SP)                  ;We need this extra word for ADWC.
                                                      011E   428 ;
                                                      011E   429 ; Initialize some locations
                                                      011E   430 ;
```

```
                     52   D4  011E    431          CLRL    R2                      ;R2 will contain number of shifts.
                     53   D4  0120    432          CLRL    R3                      ;Temporary.
                     20   B9  0122    433          BICPSW  #^X20                   ;Turn off IV so we can ASHQ
                                 0124    434 ;
                                 0124    435 ; This loop will shift OU left until MSB is a one.
                                 0124    436 ;
                                 0124    437 10$:
                     52   D7  0124    438          DECL    R2                      ;Count number of shifts.
                   0063   30  0126    439          BSBW    SHIFT_OU_LEFT           ;Shift OU left one time.
                  0F AE   95  0129    440          TSTB    15(SP)                  ;Is MSB of normalized OU set ?
                     F6   18  012C    441          BGEQ    10$                     ;No, loop again.
                                 012E    442 ;
                                 012E    443 ; Round the OU if needed
                                 012E    444 ;
                1D 6E    0E   E1  012E    445          BBC     #14, (SP), 30$          ;If the bit beyond LSB of H clear no
                                 0132    446                                  ;       rounding is needed.
                19 6E    0F   E3  0132    447          BBCS    #15, (SP), 30$          ;If this bit is clear we just need
                                 0136    448                                  ;       to set it and it is rounded.
             01 AE    80 8F   8A  0136    449          BICB2   #^X80, 1(SP)            ;Painfully round this bit pattern.
                02 AE    01   C0  013B    450          ADDL2   #1, 2(SP)               ;.....
                06 AE    00   D8  013F    451          ADWC    #0, 6(SP)               ;.....
                0A AE    00   D8  0143    452          ADWC    #0, 10(SP)              ;.....
                0E AE    00   D8  0147    453          ADWC    #0, 14(SP)              ;.....
                                 014B    454 ;
                                 014B    455 ; Normalize this bit pattern
                                 014B    456 ;
                  03 10 AE   E8  014B    457          BLBS    16(SP), 40$             ;If this bit is on, it means that we
                                 014F    458                                  ;       had an overflow on the last
                                 014F    459                                  ;       word, so it is already
                                 014F    460                                  ;       normalized.
                                 014F    461 30$:
                       003A   30  014F    462          BSBW    SHIFT_OU_LEFT           ;Normalize it.
                                 0152    463 ;
                                 0152    464 ; Put what we have cooked up in the H destination
                                 0152    465 ;
                                 0152    466 40$:
52   00000080 8F    C0  0152    467          ADDL2   #128, R2                ;exp = length of OU (128) - #of shifts.
           61   52   B0  0159    468          MOVW    R2, (R1)                ;Put in exponent.
        01 A1   40 8F   88  015C    469          BISB2   #^X40, 1(R1)            ;Set the excess bit.
        02 A1   0E AE   B0  0161    470          MOVW    14(SP), 2(R1)           ;Put in the fraction.
        04 A1   0C AE   B0  0166    471          MOVW    12(SP), 4(R1)           ;.....
        06 A1   0A AE   B0  016B    472          MOVW    10(SP), 6(R1)           ;.....
        08 A1   08 AE   B0  0170    473          MOVW    8(SP), 8(R1)            ;.....
        0A A1   06 AE   B0  0175    474          MOVW    6(SP), 10(R1)           ;.....
        0C A1   04 AE   B0  017A    475          MOVW    4(SP), 12(R1)           ;.....
        0E A1   02 AE   B0  017F    476          MOVW    2(SP), 14(R1)           ;.....
                   20   B8  0184    477          BISPSW  #^X20                   ;Turn IV back on.
                5E   12   C0  0186    478          ADDL2   #18, SP                 ;Restore SP
                   0F   BA  0189    479 50$:     POPR    #^M<R0,R1,R2,R3>
                   05   018B    480          RSB                             ;RSB back to caller.
                                 018C    481 ;
                                 018C    482 ; Subroutine to shift OU one to the left
                                 018C    483 ;
                                 018C    484 SHIFT_OU_LEFT:
                     53   94  018C    485          CLRB    R3                      ;Initialize R3
                  0B AE   95  018E    486          TSTB    11(SP)                  ;Is MSB of low quad set.
                     02   18  0191    487          BGEQ    10$                     ;No.
```

```
                                                    K 3
                     53   96  0193   488          INCB    R3                      ;Yes, remember it.
                                 0195   489 10$:
          04 AE   04 AE   01   79  0195   490          ASHQ    #1, 4(SP), 4(SP)        ;Shift low quad one time.
          0C AE   0C AE   01   79  019B   491          ASHQ    #1, 12(SP), 12(SP)      ;Shift high quad one time.
       0C AE   01   00    53   F0  01A1   492          INSV    R3, #0, #1, 12(SP)      ;If MSB of low quad was set then set
                                 01A7   493                                          ;LSB of high quad.
                               05  01A7   494          RSB                         ;Return to main routine.
```

```
                          01A8   496          .SBTTL  DBG$CVT_CVTRDQ_R1
                          01A8   497
                          01A8   498 ;++
                          01A8   499 ; FUNCTIONAL DESCRIPTION:
                          01A8   500 ;       Convert D_FLOATING to QUADWORD rounded.
                          01A8   501 ;
                          01A8   502 ;
                          01A8   503 ; CALLING SEQUENCE:
                          01A8   504 ;       DBG$CVT_CVTRDQ_R1 ( dfloat.rd.r, quad.wq.r )
                          01A8   505 ;       This is a JSB entry point.
                          01A8   506 ;
                          01A8   507 ;
                          01A8   508 ; FORMAL PARAMETERS:
                          01A8   509 ;       R0 --> dfloat   R1 --> QUADWORD
                          01A8   510 ;
                          01A8   511 ;
                          01A8   512 ; IMPLICIT INPUTS:
                          01A8   513 ;       NONE
                          01A8   514 ;
                          01A8   515 ;
                          01A8   516 ; IMPLICIT OUTPUTS:
                          01A8   517 ;       NONE
                          01A8   518 ;
                          01A8   519 ;
                          01A8   520 ; COMPLETION CODES:
                          01A8   521 ;       NONE
                          01A8   522 ;
                          01A8   523 ; SIDE EFFECTS:
                          01A8   524 ;       NONE
                          01A8   525 ;
                          01A8   526 ;--
                          01A8   527
                 00000007 01A8   528 D_OFF    = 7                                ; Offset to exponent (double)
                          01A8   529
                          01A8   530 DBG$CVT_CVTRDQ_R1 ::
                       0F BB 01A8   531          PUSHR   #^M<R0,R1,R2,R3>       ; Save these registers
                       61 7C 01AA   532          CLRD    (R1)                   ; Initialize result to zero
                    52 60 70 01AC   533          MOVD    (R0), R2               ; Grab source (reserved operand?)
                       24 13 01AF   534          BEQL    20$                    ; Branch if result is zero
                52  1000 8F A2 01B1   535          SUBW2   #32@D_OFF, R2          ; Scale down by 2**32
                  04 A1   52 6B 01B6   536          CVTRDL  R2, 4(R1)              ; Store high-order longword of result
                52  1000 8F A0 01BA   537          ADDW2   #32@D_OFF, R2          ; Scale up by 2**32
                  7E   04 A1 6E 01BF   538          CVTLD   4(R1), -(SP)           ; Convert long back to double
                       05 13 01C3   539          BEQL    10$                    ; Skip if zero
                6E  1000 8F A0 01C5   540          ADDW2   #32@D_OFF, (SP)        ; Scale up by 2**32
                    52   8E 62 01CA   541 10$:    SUBD2   (SP)+, R2              ; Subtract high-order longword
                    61   52 6B 01CD   542          CVTRDL  R2, (R1)               ; Convert rounded remainder to long
                       03 18 01D0   543          BGEQ    20$                    ; Skip if positive
                      04 A1 D7 01D2   544          DECL    4(R1)                  ; Sign-extend low-order longword
                       0F BA 01D5   545 20$:    POPR    #^M<R0,R1,R2,R3>       ; Restore these registers
                          05 01D7   546          RSB
```

```
                                                M 3
                     01D8   548          .SBTTL  DBG$CVT_CVTDH_R1
                     01D8   549
                     01D8   550 ;++
                     01D8   551 ; FUNCTIONAL DESCRIPTION:
                     01D8   552 ;       Convert D_FLOATING to H_FLOATING.
                     01D8   553 ;
                     01D8   554 ;
                     01D8   555 ; CALLING SEQUENCE:
                     01D8   556 ;       DBG$CVT_CVTDH_R1 ( dfloat.rd.r, hfloat.wh.r )
                     01D8   557 ;       This is a JSB entry point.
                     01D8   558 ;
                     01D8   559 ;
                     01D8   560 ; FORMAL PARAMETERS:
                     01D8   561 ;       R0 --> dfloat   R1 --> hfloat
                     01D8   562 ;
                     01D8   563 ;
                     01D8   564 ; IMPLICIT INPUTS:
                     01D8   565 ;       NONE
                     01D8   566 ;
                     01D8   567 ;
                     01D8   568 ; IMP' ICIT OUTPUTS:
                     01D8   569 ;       NONE
                     01D8   570 ;
                     01D8   571 ;
                     01D8   572 ; COMPLETION CODES:
                     01D8   573 ;       NONE
                     01D8   574 ;
                     01D8   575 ; SIDE EFFECTS:
                     01D8   576 ;       OPCODE RESERVED TO DIGITAL error is possible, see ENVIRONMENT.
                     01D8   577 ;
                     01D8   578 ;--
                     01D8   579
                     01D8   580 DBG$CVT_CVTDH_R1 ::
61   60 32FD         01D8   581          CVTDH   (R0), (R1)          ;Convert D_FLOATING to H_FLOATING
         05          01DC   582          RSB
```

```
                     01DD    584          .SBTTL  DBG$CVT_CVTRHL_R1
                     01DD    585
                     01DD    586 ;++
                     01DD    587 ; FUNCTIONAL DESCRIPTION:
                     01DD    588 ;       Convert H_FLOATING to LONGWORD rounded.
                     01DD    589 ;
                     01DD    590 ;
                     01DD    591 ; CALLING SEQUENCE:
                     01DD    592 ;       DBG$CVT_CVTRHL_R1 ( hfloat.rh.r, long.wl.r )
                     01DD    593 ;       This is a JSB entry point.
                     01DD    594 ;
                     01DD    595 ;
                     01DD    596 ; FORMAL PARAMETERS:
                     01DD    597 ;       R0 --> hfloat   R1 --> long
                     01DD    598 ;
                     01DD    599 ;
                     01DD    600 ; IMPLICIT INPUTS:
                     01DD    601 ;       NONE
                     01DD    602 ;
                     01DD    603 ;
                     01DD    604 ; IMPLICIT OUTPUTS:
                     01DD    605 ;       NONE
                     01DD    606 ;
                     01DD    607 ;
                     01DD    608 ; COMPLETION CODES:
                     01DD    609 ;       NONE
                     01DD    610 ;
                     01DD    611 ; SIDE EFFECTS:
                     01DD    612 ;       OPCODE RESERVED TO DIGITAL error is possible, see ENVIRONMENT.
                     01DD    613 ;
                     01DD    614 ;-
                     01DD    615 DBG$CVT_CVTRHL_R1 ::
           61    60 6BFD     01DD    616          CVTRHL  (R0), (R1)              ;Convert H_FLOATING to LONGWORD rounded
                        05   01E1    617          RSB
```

```
                                01E2    619          .SBTTL  DBG$CVT_CVTRHQ_R1
                                01E2    620 ;++
                                01E2    621 ; FUNCTIONAL DESCRIPTION:
                                01E2    622 ;       Convert H_FLOATING to QUADWORD rounded.
                                01E2    623 ;
                                01E2    624 ;
                                01E2    625 ; CALLING SEQUENCE:
                                01E2    626 ;       DBG$CVT_CVTRHQ_R1 ( hfloat.rh.r, quad.wq.r )
                                01E2    627 ;       This is a JSB entry point.
                                01E2    628 ;
                                01E2    629 ;
                                01E2    630 ; FORMAL PARAMETERS:
                                01E2    631 ;       R0 --> hfloat   R1 --> quad
                                01E2    632 ;
                                01E2    633 ;
                                01E2    634 ; IMPLICIT INPUTS:
                                01E2    635 ;       NONE
                                01E2    636 ;
                                01E2    637 ;
                                01E2    638 ; IMPLICIT OUTPUTS:
                                01E2    639 ;       NONE
                                01E2    640 ;
                                01E2    641 ;
                                01E2    642 ; COMPLETION CODES:
                                01E2    643 ;       NONE
                                01E2    644 ;
                                01E2    645 ; SIDE EFFECTS:
                                01E2    646 ;
                                01E2    647 ;
                                01E2    648 ;--
                                01E2    649 DBG$CVT_CVTRHQ_R1 ::
                    0F    BB    01E2    650          PUSHR   #^M<R0,R1,R2,R3>        ; Save these registers
                    61    7C    01E4    651          CLRQ    (R1)                    ; Initialize result to zero
                    53    D4    01E6    652          CLRL    R3                      ; Initialize rounding bit
                                01E8    653 ;
                                01E8    654 ;        Find the exponent
                                01E8    655 ;
          52    60    0F    00  EF    01E8    656          EXTZV   #0, #15, (R0), R2       ; Extract the exponent
                    0D    13  01ED    657          BEQL    5$                      ; Call the number zero if expo. is zero
           52   00004000 8F    C2    01EF    658          SUBL2   #^X4000, R2             ; Take out the excess
                    04    19  01F6    659          BLSS    5$                      ; If negative then Quad is zero
                    05    14  01F8    660          BGTR    10$                     ; Is exponent zero ?
                    61    D6  01FA    661          INCL    (R1)                    ; Yes, call it a one
                                01FC    662 5$:
                  007D    31  01FC    663          BRW     50$                     ; Go to the finish line
                                01FF    664 ;
                                01FF    665 ;        Check for overflow.  If exponent is 63, find rounding bit before we
                                01FF    666 ;        lose it.
                                01FF    667 ;
                                01FF    668 10$:
                52    3F    C2  01FF    669          SUBL2   #63, R2                 ; Bring '.' to right of bit 0
                    1E    19  0202    670          BLSS    20$                     ; If exponent < 63, skip this
                    0F    14  0204    671          BGTR    15$                     ; If exponent > 63, we have IV
    53    08 A0   01    01  EF    0206    672          EXTZV   #1, #1, 8(R0), R3       ; Exponent is 63, R3 is the round bit
           53   80000000 8F    CA    020C    673          BICL2   #^X80000000, R3         ; Turn on the 31st bit of R3 to flag
                                0213    674                                          ; that the round bit has been determined
                    0D    11  0213    675          BRB     20$                     ; Cont.
```

```
                                0215   676 15$:
          0000047C 8F   DD      0215   677          PUSHL   #SS$_INTOVF              ; Signal IV
00000000'GF    01   FB      021B   678          CALLS   #1, G^LIB$SIGNAL        ;
                                0222   679 ;
                                0222   680 ;       Move the fraction to the QUAD, and put in a positive sign and MSB
                                0222   681 ;
                                0222   682 20$:
       61    08 A0   B0      0222   683          MOVW    8(R0), (R1)             ; Move the fraction to QUAD
    02 A1    06 A0   B0      0226   684          MOVW    6(R0), 2(R1)            ;
    04 A1    04 A0   B0      022B   685          MOVW    4(R0), 4(R1)            ;
    06 A1    02 A0   B0      0230   686          MOVW    2(R0), 6(R1)            ;
 61    61    FE 8F   79      0235   687          ASHQ    #-2, (R1), (R1)         ; Make room for sign and MSB
    07 A1    C0 8F   8A      023A   688          BICB2   #^XC0, 7(R1)            ; Make sure two high bits are off
    07 A1    40 8F   88      023F   689          BISB2   #^X40, 7(R1)            ; Sign is '+', MSB is 1
                                0244   690 ;
                                0244   691 ;       Find rounding bit if not already done
                                0244   692 ;
                      53    D5  0244   693          TSTL    R3                      ; Is rounding bit already found ?
                      0A    14  0246   694          BGTR    30$                     ; Yes
                53    52    CE  0248   695          MNEGL   R2, R3                  ; R3 will contain rounding bit
                      53    D7  024B   696          DECL    R3                      ; Look at the right of decimal point
 53    61    01    53    EF  024D   697          EXTZV   R3, #1, (R1), R3        ; This is the rounding bit
                                0252   698 ;
                                0252   699 ;       Shift the QAUD R2 times
                                0252   700 ;
                                0252   701 30$:
       61    61    52    79  0252   702          ASHQ    R2, (R1), (R1)          ; Shift the QUAD R2 times
                      53    95  0256   703          TSTB    R3                      ; Do we need to round
                      0B    13  0258   704          BEQL    40$                     ; No
             52       61    D0  025A   705          MOVL    (R1), R2                ; Do the rounding
       61       01    A2    9E  025D   706          MOVAB   1(R2), (R1)             ; ...
       04 A1          00    D8  0261   707          ADWC    #0, 4(R1)               ; ...
                      60    B5  0265   708 40$:     TSTW    (R0)                    ; Is the H negative ?
                      13    14  0267   709          BGTR    50$                     ; No
       61             61    D2  0269   710          MCOML   (R1), (R1)              ; Negate the QUAD
 04 A1          04    A1    D2  026C   711          MCOML   4(R1), 4(R1)            ; ...
             50       61    D0  0271   712          MOVL    (R1), R0                ; ...
       61       01    A0    9E  0274   713          MOVAB   1(R0), (R1)             ; ...
       04 A1          00    D8  0278   714          ADWC    #0, 4(R1)               ; ...
                                027C   715 50$:
                      0F    BA  027C   716          POPR    #^M<R0,R1,R2,R3>        ; Restore these registers
                            05  027E   717          RSB
```

```
                                    027F    719          .SBTTL  DBG$CVT_CVTRHO_R1
                                    027F    720 ;++
                                    027F    721 ; FUNCTIONAL DESCRIPTION:
                                    027F    722 ;       Convert H_FLOATING to OCTAWORD rounded.
                                    027F    723 ;
                                    027F    724 ;
                                    027F    725 ; CALLING SEQUENCE:
                                    027F    726 ;       DBG$CVT_CVTRHO_R1 ( hfloat.rh.r, octa.wo.r )
                                    027F    727 ;       This is a JSB entry point.
                                    027F    728 ;
                                    027F    729 ;
                                    027F    730 ; FORMAL PARAMETERS:
                                    027F    731 ;       R0 --> hfloat   R1 --> octa
                                    027F    732 ;
                                    027F    733 ;
                                    027F    734 ; IMPLICIT INPUTS:
                                    027F    735 ;       NONE
                                    027F    736 ;
                                    027F    737 ;
                                    027F    738 ; IMPLICIT OUTPUTS:
                                    027F    739 ;       NONE
                                    027F    740 ;
                                    027F    741 ;
                                    027F    742 ; COMPLETION CODES:
                                    027F    743 ;       NONE
                                    027F    744 ;
                                    027F    745 ; SIDE EFFECTS:
                                    027F    746 ;       OPCODE RESERVED TO DIGITAL error is possible, see ENVIRONMENT.
                                    027F    747 ;
                                    027F    748 ;--
                00000004            027F    749 O_LEN   = 4             ; Length in longwords (octaword)
                                    027F    750
                                    027F    751 DBG$CVT_CVTRHO_R1 ::
                                    027F    752 ;
                                    027F    753 ; Initialization
                                    027F    754 ;
                         3F    BB   027F    755         PUSHR   #^M<R0,R1,R2,R3,R4,R5>  ; Save these registers
                         61    7C   0281    756         CLRQ    (R1)                    ; Initialize result to zero
                      08 A1    7C   0283    757         CLRQ    8(R1)                   ; Initialize result to zero
                   7E    60 70FD   0286    758         MOVH    (R0), -(SP)             ; Grab source (reserved operand?)
                         43    13   028A    759         BEQL    30$                     ; Branch if result is zero
                                    028C    760 ;
                                    028C    761 ; The source is not zero; initialize other registers
                                    028C    762 ;
                   51    10    C0   028C    763         ADDL2   #O_LEN*4, R1            ; Address past octaword
                   54    51    D0   028F    764         MOVL    R1, R4                  ; Save this address
                   55    03    D0   0292    765         MOVL    #O_LEN-1, R5            ; Loop counter
                                    0295    766 ;
                                    0295    767 10$:    ; Loop for each longword of destination
                                    0295    768 ;
               52  55    05    78   0295    769         ASHL    #5, R5, R2              ; Scaling amount (2**5 = 32)
                   53    51    D0   0299    770         MOVL    R1, R3                  ; Address where we start decrementing
                   6E    52    A2   029C    771         SUBW2   R2, (SP)                ; Scale down by 2**(32*n)
                   71    6E 6BFD   029F    772         CVTRHL  (SP), -(R1)             ; Store high-order longword of result
                   6E    52    A0   02A3    773         ADDW2   R2, (SP)                ; Scale up by 2**(32*n)
                   7E    61 6EFD   02A6    774         CVTLH   (R1), -(SP)             ; Convert long back to double
                         1C    13   02AA    775         BEQL    20$                     ; Skip if zero
```

```
                     10   11  02AC   776          BRB     13$                     ; Jump into decrement loop
                                 02AE   777          ;
                                 02AE   778          ; If the longword was negative, we must sign-extend the result
                                 02AE   779          ;
          80000000 8F    63    D1  02AE   780 11$:     CMPL    (R3), #^X80000000       ; Compare with largest negative number
                         05    13  02B5   781          BEQL    12$                     ; If not equal, do a simple decrement
                   83    63    D2  02B7   782          MCOML   (R3), (R3)+             ; Subtract one and consider negative
                         04    11  02BA   783          BRB     14$                     ; Stay in loop, as result is negative
                         83    D7  02BC   784 12$:     DECL    (R3)+                   ; Decrement this longword
                         05    18  02BE   785 13$:     BGEQ    15$                     ; Jump out of loop if positive
                   54    53    D1  02C0   786 14$:     CMPL    R3, R4                  ; See if we are done decrementing
                         E9    1F  02C3   787          BLSSU   11$                     ; If not past octaword, keep going
                                 02C5   788          ;
                                 02C5   789          ; Scale up the longword (if not zero), and subtract from stack temp
                                 02C5   790          ;
                   6E    52    A0  02C5   791 15$:     ADDW2   R2, (SP)                ; Scale up by 2**(32*n)
                   6E    8E 62FD  02C8   792 20$:     SUBH2   (SP)+, (SP)             ; Subtract high-order longword
                                 02CC   793          ;
                                 02CC   794          ; Decrease loop counter, and continue, if more longwords required
                                 02CC   795          ;
                     C6 55    F4  02CC   796          SOBGEQ  R5, 10$                 ; Continue looping
                                 02CF   797          ;
                                 02CF   798 30$:     ; Clean up the stack, and restore registers
                                 02CF   799          ;
                        8E 7CFD  02CF   800          CLRH    (SP)+                   ; Pop huge from the stack
                        3F    BA  02D2   801          POPR    #^M<R0,R1,R2,R3,R4,R5>  ; Restore these registers
                              05  02D4   802          RSB                             ; Return from subroutine
                                 02D5   803
```

```
                     02D5    805          .SBTTL  DBG$CVT_CVTHF_R1
                     02D5    806
                     02D5    807 ;++
                     02D5    808 ; FUNCTIONAL DESCRIPTION:
                     02D5    809 ;       Convert H_FLOATING to FLOATING
                     02D5    810 ;
                     02D5    811 ;
                     02D5    812 ; CALLING SEQUENCE:
                     02D5    813 ;       DBG$CVT_CVTHF_R1 (hfloat.rh.r, float.wf.r )
                     02D5    814 ;       This is a JSB entry point.
                     02D5    815 ;
                     02D5    816 ;
                     02D5    817 ; FORMAL PARAMETERS:
                     02D5    818 ;       R0 --> hfloat   R1 --> float
                     02D5    819 ;
                     02D5    820 ;
                     02D5    821 ; IMPLICIT INPUTS:
                     02D5    822 ;       NONE
                     02D5    823 ;
                     02D5    824 ;
                     02D5    825 ; IMPLICIT OUTPUTS:
                     02D5    826 ;       NONE
                     02D5    827 ;
                     02D5    828 ;
                     02D5    829 ; COMPLETION CODES:
                     02D5    830 ;       NONE
                     02D5    831 ;
                     02D5    832 ; SIDE EFFECTS:
                     02D5    833 ;       OPCODE RESERVED TO DIGITAL error is possible, see ENVIRONMENT.
                     02D5    834 ;
                     02D5    835 ;--
                     02D5    836 DBG$CVT_CVTHF_R1 ::
61       60 F6FD     02D5    837          CVTHF   (R0), (R1)              ;Convert H_FLOATING to FLOAT
                       05    02D9    838          RSB
```

```
                        .SBTTL  DBG$CVT_CVTHD_R1

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

```
                     02DF    875          .SBTTL  DBG$CVT_CVTHG_R1
                     02DF    876
                     02DF    877 ;++
                     02DF    878 ; FUNCTIONAL DESCRIPTION:
                     02DF    879 ;       Convert H_FLOATING to G_FLOATING
                     02DF    880 ;
                     02DF    881 ;
                     02DF    882 ; CALLING SEQUENCE:
                     02DF    883 ;       DBG$CVT_CVTHG_R1 ( hfloat.rh.r, gfloat.wg.r )
                     02DF    884 ;       This is a JSB entry point.
                     02DF    885 ;
                     02DF    886 ;
                     02DF    887 ; FORMAL PARAMETERS:
                     02DF    888 ;       R0 --> hfloat   R1 --> gfloat
                     02DF    889 ;
                     02DF    890 ;
                     02DF    891 ; IMPLICIT INPUTS:
                     02DF    892 ;       NONE
                     02DF    893 ;
                     02DF    894 ;
                     02DF    895 ; IMPLICIT OUTPUTS:
                     02DF    896 ;       NONE
                     02DF    897 ;
                     02DF    898 ;
                     02DF    899 ; COMPLETION CODES:
                     02DF    900 ;       NONE
                     02DF    901 ;
                     02DF    902 ; SIDE EFFECTS:
                     02DF    903 ;       OPCODE RESERVED TO DIGITAL error is possible, see ENVIRONMENT.
                     02DF    904 ;
                     02DF    905 ;--
                     02DF    906 DBG$CVT_CVTHG_R1 ::
61     60 76FD       02DF    907          CVTHG   (R0), (R1)              ;Convert H_FLOATING to G_FLOATING
          05         02E3    908          RSB
```

```
                     02E4    910          .SBTTL  DBG$CVT_CVTGH_R1
                     02E4    911
                     02E4    912 ;++
                     02E4    913 ; FUNCTIONAL DESCRIPTION:
                     02E4    914 ;       Convert G_FLOATING to H_FLOATING
                     02E4    915 ;
                     02E4    916 ;
                     02E4    917 ; CALLING SEQUENCE:
                     02E4    918 ;       DBG$CVT_CVTGH_R1 ( gfloat.rg.r, hfloat.wh.r )
                     02E4    919 ;       This is a JSB entry point.
                     02E4    920 ;
                     02E4    921 ;
                     02E4    922 ; FORMAL PARAMETERS:
                     02E4    923 ;       R0 --> gfloat   R1 --> hfloat
                     02E4    924 ;
                     02E4    925 ;
                     02E4    926 ; IMPLICIT INPUTS:
                     02E4    927 ;       NONE
                     02E4    928 ;
                     02E4    929 ;
                     02E4    930 ; IMPLICIT OUTPUTS:
                     02E4    931 ;       NONE
                     02E4    932 ;
                     02E4    933 ;
                     02E4    934 ; COMPLETION CODES:
                     02E4    935 ;       NONE
                     02E4    936 ;
                     02E4    937 ; SIDE EFFECTS:
                     02E4    938 ;       OPCODE RESERVED TO DIGITAL error is possible, see ENVIRONMENT.
                     02E4    939 ;
                     02E4    940 ;--
                     02E4    941 DBG$CVT_CVTGH_R1 ::
  61    60 56FD      02E4    942          CVTGH   (R0), (R1)              ;Convert G_FLOATING TO H_FLOATING
        05           02E8    943          RSB
```

```
                                02E9   945          .SBTTL  DBG$CVT_SCALE_OU_UP_BY_10_R1
                                02E9   946
                                02E9   947 ;++
                                02E9   948 ; FUNCTIONAL DESCRIPTION:
                                02E9   949 ;       Scale an unsigned OCTAWORD up by 10.
                                02E9   950 ;
                                02E9   951 ;
                                02E9   952 ; CALLING SEQUENCE:
                                02E9   953 ;       DBG$CVT_SCALE_OU_UP_BY_10_R1 ( octa.mo.r )
                                02E9   954 ;       This is a JSB entry point.
                                02E9   955 ;
                                02E9   956 ;
                                02E9   957 ; FORMAL PARAMETERS:
                                02E9   958 ;       R0 --> octa
                                02E9   959 ;
                                02E9   960 ;
                                02E9   961 ; IMPLICIT INPUTS:
                                02E9   962 ;       NONE
                                02E9   963 ;
                                02E9   964 ;
                                02E9   965 ; IMPLICIT OUTPUTS:
                                02E9   966 ;       NONE
                                02E9   967 ;
                                02E9   968 ;
                                02E9   969 ; COMPLETION CODES:
                                02E9   970 ;       NONE
                                02E9   971 ;
                                02E9   972 ; SIDE EFFECTS:
                                02E9   973 ;
                                02E9   974 ;       NONE
                                02E9   975 ;--
                                02E9   976 DBG$CVT_SCALE_OU_UP_BY_10_R1 ::
                          1F    BB  02E9   977          PUSHR   #^M<R0,R1,R2,R3,R4>
                    54    04    D0  02EB   978          MOVL    #4, R4                  ; Do four longwords
                          53    D4  02EE   979          CLRL    R3                      ; Clear the 'carry'
51  53    60          0A    7A  02F0   980 10$:     EMUL    #10, (R0), R3, R1       ; Multiply low-order longwords
                          03    18  02F5   981          BGEQ    20$                     ; Make sure low-order longword
                    52    0A    C0  02F7   982          ADDL2   #10, R2                 ;    is seen as unsigned
                    80    51    D0  02FA   983 20$:     MOVL    R1, (R0)+               ; Store low longword
                    53    52    D0  02FD   984          MOVL    R2, R3                  ; Save the 'carry'
                      ED  54    F5  0300   985          SOBGTR  R4, 10$                 ; Continue looping
                          53    D5  0303   986          TSTL    R3                      ; Check for overflow
                          0E    13  0305   987          BEQL    30$                     ; Branch if no overflow
 7E     0000047C          8F    DD  0307   988          MOVL    #SS$_INTOVF, -(SP)      ; Integer overflow
 00000000'GF              01    FB  030E   989          CALLS   #1, G^LIB$SIGNAL        ; Signal the error
                          1F    BA  0315   990 30$:     POPR    #^M<R0,R1,R2,R3,R4>
                          05    05  0317   991          RSB                             ; Return
                                0318   992
```

```
                        0318   994          .SBTTL  DBG$CVT_SCALE_OU_DOWN_BY_10_R1
                        0318   995
                        0318   996 ;++
                        0318   997 ; FUNCTIONAL DESCRIPTION:
                        0318   998 ;       Scales down an OCTAWORD by 10.
                        0318   999 ;
                        0318  1000 ;
                        0318  1001 ; CALLING SEQUENCE:
                        0318  1002 ;       DBG$CVT_SCALE_OU_DOWN_BY_10_R1 ( octa.mo.r )
                        0318  1003 ;       This is a JSB entry point.
                        0318  1004 ;
                        0318  1005 ;
                        0318  1006 ; FORMAL PARAMETERS:
                        0318  1007 ;       R0 --> octa
                        0318  1008 ;
                        0318  1009 ;
                        0318  1010 ; IMPLICIT INPUTS:
                        0318  1011 ;       NONE
                        0318  1012 ;
                        0318  1013 ;
                        0318  1014 ; IMPLICIT OUTPUTS:
                        0318  1015 ;       NONE
                        0318  1016 ;
                        0318  1017 ;
                        0318  1018 ; COMPLETION CODES:
                        0318  1019 ;       NONE
                        0318  1020 ;
                        0318  1021 ; SIDE EFFECTS:
                        0318  1022 ;       NONE
                        0318  1023 ;
                        0318  1024 ;--
                        0318  1025 DBG$CVT_SCALE_OU_DOWN_BY_10_R1::
                    1F  BB  0318  1026          PUSHR   #^M<R0,R1,R2,R3,R4>
                54  03  D0  031A  1027          MOVL    #3, R4                  ; Do four longwords
                    52  D4  031D  1028          CLRL    R2                      ; Clear high longword of quad
            51  6044  D0  031F  1029 10$:      MOVL    (R0)[R4], R1            ; Grab low longword of quad
52  6044    51  0A  7B  0323  1030          EDIV    #10, R1, (R0)[R4], R2   ; Do a divide
                52  05  D1  0329  1031 30$:      CMPL    #5, R2                  ; See if remainder too large
                    0D  1A  032C  1032          BGTRU   50$                     ;    for the next EDIV
                    08  15  032E  1033          BLEQ    40$                     ; Simply make remainder smaller
                52  0A  C0  0330  1034          ADDL2   #10, R2                 ; Make remainder positive
                6044  D7  0333  1035          DECL    (R0)[R4]                ; Make quotient smaller
                    F1  11  0336  1036          BRB     30$                     ; Join the looping code
                52  0A  C2  0338  1037 40$:      SUBL2   #10, R2                 ; Make remainder smaller
                  E1 54  F4  033B  1038 50$:      SOBGEQ  R4, 10$                 ; Continue looping
                    1F  BA  033E  1039          POPR    #^M<R0,R1,R2,R3,R4>
                        05  0340  1040          RSB                             ; Return
```

```
                     0341  1042          .SBTTL  DBG$CVT_SCALE_OU_UP_BY_2_R1
                     0341  1043
                     0341  1044 ;++
                     0341  1045 ; FUNCTIONAL DESCRIPTION:
                     0341  1046 ;       Scale an unsigned OCTAWORD up by 2.
                     0341  1047 ;
                     0341  1048 ;
                     0341  1049 ; CALLING SEQUENCE:
                     0341  1050 ;       DBG$CVT_SCALE_OU_UP_BY_2_R1 ( octa.mo.r )
                     0341  1051 ;       This is a JSB entry point.
                     0341  1052 ;
                     0341  1053 ;
                     0341  1054 ; FORMAL PARAMETERS:
                     0341  1055 ;       R0 --> octa
                     0341  1056 ;
                     0341  1057 ;
                     0341  1058 ; IMPLICIT INPUTS:
                     0341  1059 ;       NONE
                     0341  1060 ;
                     0341  1061 ;
                     0341  1062 ; IMPLICIT OUTPUTS:
                     0341  1063 ;       NONE
                     0341  1064 ;
                     0341  1065 ;
                     0341  1066 ; COMPLETION CODES:
                     0341  1067 ;       NONE
                     0341  1068 ;
                     0341  1069 ; SIDE EFFECTS:
                     0341  1070 ;
                     0341  1071 ;       NONE
                     0341  1072 ;--
                     0341  1073 DBG$CVT_SCALE_OU_UP_BY_2_R1 ::
                 1F  BB  0341  1074          PUSHR   #^M<R0,R1,R2,R3,R4>
           54    04  D0  0343  1075          MOVL    #4, R4                  ; Do four longwords
                 53  D4  0346  1076          CLRL    R3                      ; Clear the 'carry'
     51    53    60  02  7A  0348  1077 10$:  EMUL    #2, (R0), R3, R1        ; Multiply low-order longwords
                 03  18  034D  1078          BGEQ    20$                     ; Make sure low-order longword
           52    02  C0  034F  1079          ADDL2   #2, R2                  ;    is seen as unsigned
           80    51  D0  0352  1080 20$:     MOVL    R1, (R0)+               ; Store low longword
           53    52  D0  0355  1081          MOVL    R2, R3                  ; Save the 'carry'
             ED  54  F5  0358  1082          SOBGTR  R4, 10$                 ; Continue looping
                 53  D5  035B  1083          TSTL    R3                      ; Check for overflow
                 0E  13  035D  1084          BEQL    30$                     ; Branch if no overflow
      7E   0000047C  8F  DD  035F  1085          MOVL    #SS$_INTOVF, -(SP)      ; Integer overflow
      00000000'GF    01  FB  0366  1086          CALLS   #1, G^LIB$SIGNAL        ; Signal the error
                 1F  BA  036D  1087 30$:     POPR    #^M<R0,R1,R2,R3,R4>
                 05  036F  1088          RSB                             ; Return
                     0370  1089
```

```
                                0370  1091          .SBTTL  DBG$CVT_SCALE_OU_DOWN_BY_2_R1
                                0370  1092
                                0370  1093 ;++
                                0370  1094 ; FUNCTIONAL DESCRIPTION:
                                0370  1095 ;       Scales down an OCTAWORD by 2.
                                0370  1096 ;
                                0370  1097 ;
                                0370  1098 ; CALLING SEQUENCE:
                                0370  1099 ;       DBG$CVT_SCALE_OU_DOWN_BY_2_R1 ( octa.mo.r )
                                0370  1100 ;       This is a JSB entry point.
                                0370  1101 ;
                                0370  1102 ;
                                0370  1103 ; FORMAL PARAMETERS:
                                0370  1104 ;       R0 --> octa
                                0370  1105 ;
                                0370  1106 ;
                                0370  1107 ; IMPLICIT INPUTS:
                                0370  1108 ;       NONE
                                0370  1109 ;
                                0370  1110 ;
                                0370  1111 ; IMPLICIT OUTPUTS:
                                0370  1112 ;       NONE
                                0370  1113 ;
                                0370  1114 ;
                                0370  1115 ; COMPLETION CODES:
                                0370  1116 ;       NONE
                                0370  1117 ;
                                0370  1118 ; SIDE EFFECTS:
                                0370  1119 ;       NONE
                                0370  1120 ;
                                0370  1121 ;--
                                0370  1122 DBG$CVT_SCALE_OU_DOWN_BY_2_R1::
                      1F    BB  0370  1123          PUSHR   #^M<R0,R1,R2,R3,R4>
                   54   03    D0  0372  1124          MOVL    #3, R4                  ; Do four longwords
                        52    D4  0375  1125          CLRL    R2                      ; Clear high longword of quad
                51      6044    D0  0377  1126 10$:     MOVL    (R0)[R4], R1            ; Grab low longword of quad
52      6044    51      02    7B  037B  1127          EDIV    #2, R1, (R0)[R4], R2    ; Do a divide
                52      01    D1  0381  1128 30$:     CMPL    #1, R2                  ; See if remainder too large
                        0D    1A  0384  1129          BGTRU   50$                     ;     for the next EDIV
                        08    15  0386  1130          BLEQ    40$                     ; Simply make remainder smaller
                52      02    C0  0388  1131          ADDL2   #2, R2                  ; Make remainder positive
                   6044     D7  038B  1132          DECL    (R0)[R4]                ; Make quotient smaller
                        F1    11  038E  1133          BRB     30$                     ; Join the looping code
                52      02    C2  0390  1134 40$:     SUBL2   #2, R2                  ; Make remainder smaller
                  E1    54    F4  0393  1135 50$:     SOBGEQ  R4, 10$                 ; Continue looping
                        1F    BA  0396  1136          POPR    #^M<R0,R1,R2,R3,R4>
                              05  0398  1137          RSB                             ; Return
                                  0399  1138
```

```
                     0399  1140          .SBTTL  DBG$CVT_MULD2_R1
                     0399  1141
                     0399  1142 ;++
                     0399  1143 ; FUNCTIONAL DESCRIPTION:
                     0399  1144 ;       multiply two DOUBLE floating values.
                     0399  1145 ;
                     0399  1146 ;
                     0399  1147 ; CALLING SEQUENCE:
                     0399  1148 ;       DBG$CVT_MULD2_R1 ( mulr.rd.r, prod.md.r )
                     0399  1149 ;       This is a JSB entry point.
                     0399  1150 ;
                     0399  1151 ;
                     0399  1152 ; FORMAL PARAMETERS:
                     0399  1153 ;       R0 --> mulr     R1 --> prod
                     0399  1154 ;
                     0399  1155 ;
                     0399  1156 ; IMPLICIT INPUTS:
                     0399  1157 ;       NONE
                     0399  1158 ;
                     0399  1159 ;
                     0399  1160 ; IMPLICIT OUTPUTS:
                     0399  1161 ;       NONE
                     0399  1162 ;
                     0399  1163 ;
                     0399  1164 ; COMPLETION CODES:
                     0399  1165 ;       NONE
                     0399  1166 ;
                     0399  1167 ; SIDE EFFECTS:
                     0399  1168 ;       NONE
                     0399  1169 ;
                     0399  1170 ;--
                     0399  1171 DBG$CVT_MULD2_R1 ::
 61    60    64      0399  1172          MULD2   (R0), (R1)              ;Multiply two D_FLOATING
                05    039C  1173          RSB
```

```
                                B 5
039D  1175          .SBTTL  DBG$CVT_DIVD2_R1
039D  1176
039D  1177 ;++
039D  1178 ; FUNCTIONAL DESCRIPTION:
039D  1179 ;       Divide two DOUBLE floating values.
039D  1180 ;
039D  1181 ;
039D  1182 ; CALLING SEQUENCE:
039D  1183 ;       DBG$CVT_DIVD2_R1 ( divr.rd.r, quo.md.r )
039D  1184 ;       This is a JSB entry point.
039D  1185 ;
039D  1186 ;
039D  1187 ; FORMAL PARAMETERS:
039D  1188 ;       R0 --> divr     R1 --> quo
039D  1189 ;
039D  1190 ;
039D  1191 ; IMPLICIT INPUTS:
039D  1192 ;       NONE
039D  1193 ;
039D  1194 ;
039D  1195 ; IMPLICIT OUTPUTS:
039D  1196 ;       NONE
039D  1197 ;
039D  1198 ;
039D  1199 ; COMPLETION CODES:
039D  1200 ;       NONE
039D  1201 ;
039D  1202 ; SIDE EFFECTS:
039D  1203 ;       NONE
039D  1204 ;
039D  1205 ;--
039D  1206 DBG$CVT_DIVD2_R1 ::
61   60   66  039D  1207          DIVD2   (R0), (R1)              ;Divide two D_FLOATING
              05  03A0  1208          RSB
```

```
                                                C 5
                     03A1  1210          .SBTTL  DBG$CVT_MULH2_R1
                     03A1  1211
                     03A1  1212 ;++
                     03A1  1213 ; FUNCTIONAL DESCRIPTION:
                     03A1  1214 ;       multiply two H_FLOATING floating values.
                     03A1  1215 ;
                     03A1  1216 ;
                     03A1  1217 ; CALLING SEQUENCE:
                     03A1  1218 ;       DBG$CVT_MULH2_R1 ( mulr.rh.r, prod.mh.r )
                     03A1  1219 ;       This is a JSB entry point.
                     03A1  1220 ;
                     03A1  1221 ;
                     03A1  1222 ; FORMAL PARAMETERS:
                     03A1  1223 ;       R0 --> mulr     R1 --> prod
                     03A1  1224 ;
                     03A1  1225 ;
                     03A1  1226 ; IMPLICIT INPUTS:
                     03A1  1227 ;       NONE
                     03A1  1228 ;
                     03A1  1229 ;
                     03A1  1230 ; IMPLICIT OUTPUTS:
                     03A1  1231 ;       NONE
                     03A1  1232 ;
                     03A1  1233 ;
                     03A1  1234 ; COMPLETION CODES:
                     03A1  1235 ;       NONE
                     03A1  1236 ;
                     03A1  1237 ; SIDE EFFECTS:
                     03A1  1238 ;
                     03A1  1239 ;       OPCODE RESERVED TO DIGITAL error is possible, see ENVIRONMENT.
                     03A1  1240 ;
                     03A1  1241 ;--
                     03A1  1242 DBG$CVT_MULH2_R1 ::
61   60 64FD         03A1  1243          MULH2   (R0), (R1)              ;Multiply two H_FLOATING
         05          03A5  1244          RSB
```

```
                     03A6  1246          .SBTTL  DBG$CVT_DIVH2_R1
                     03A6  1247
                     03A6  1248 ;++
                     03A6  1249 ; FUNCTIONAL DESCRIPTION:
                     03A6  1250 ;       Divide two H_FLOATING floating values.
                     03A6  1251 ;
                     03A6  1252 ;
                     03A6  1253 ; CALLING SEQUENCE:
                     03A6  1254 ;       DBG$CVT_DIVH2_R1 ( divr.rh.r, quo.mh.r )
                     03A6  1255 ;       This is a JSB entry point.
                     03A6  1256 ;
                     03A6  1257 ;
                     03A6  1258 ; FORMAL PARAMETERS:
                     03A6  1259 ;       R0 --> divr     R1 --> quo
                     03A6  1260 ;
                     03A6  1261 ;
                     03A6  1262 ; IMPLICIT INPUTS:
                     03A6  1263 ;       NONE
                     03A6  1264 ;
                     03A6  1265 ;
                     03A6  1266 ; IMPLICIT OUTPUTS:
                     03A6  1267 ;       NONE
                     03A6  1268 ;
                     03A6  1269 ;
                     03A6  1270 ; COMPLETION CODES:
                     03A6  1271 ;       NONE
                     03A6  1272 ;
                     03A6  1273 ; SIDE EFFECTS:
                     03A6  1274 ;
                     03A6  1275 ;       OPCODE RESERVED TO DIGITAL error is possible, see ENVIRONMENT.
                     03A6  1276 ;
                     03A6  1277 ;--
                     03A6  1278 DBG$CVT_DIVH2_R1 ::
61    60 66FD        03A6  1279          DIVH2   (R0), (R1)              ;Divide two H_FLOATING
         05          03AA  1280          RSB
```

```
                          1282          .SBTTL  DBG$CVT_ASHP_R1
                          1283
                          1284 ;++
                          1285 ; FUNCTIONAL DESCRIPTION:
                          1286 ;       Scale a packed decimal number.
                          1287 ;
                          1288 ;
                          1289 ; CALLING SEQUENCE:
                          1290 ;       DBG$CVT_ASHP_R1 (cnt.rw.r, srclen.rw.r, srcaddr.ra.r,
                          1291 ;                        round.rb.r, dstlen.rw.r, dstaddr.ra.r )
                          1292 ;       This is a JSB entry point.
                          1293 ;
                          1294 ;
                          1295 ; FORMAL PARAMETERS:
                          1296 ;       R0 --> cnt      R1 = srclen     R2 --> srcaddr  R3 = round
                          1297 ;       R4 --> dstlen   R5 = dstaddr
                          1298 ;
                          1299 ;
                          1300 ; IMPLICIT INPUTS:
                          1301 ;       NONE
                          1302 ;
                          1303 ;
                          1304 ; IMPLICIT OUTPUTS:
                          1305 ;       NONE
                          1306 ;
                          1307 ;
                          1308 ; COMPLETION CODES:
                          1309 ;       NONE
                          1310 ;
                          1311 ; SIDE EFFECTS:
                          1312 ;       NONE
                          1313 ;
                          1314 ;--
                          1315 DBG$CVT_ASHP_R1 ::
                    3F BB 03AB 1316          PUSHR   #^M<R0,R1,R2,R3,R4,R5>
65  64  63  62  61   60 F8 03AD 1317          ASHP    (R0), (R1), (R2), (R3), (R4), (R5)
                    3F BA 03B4 1318          POPR    #^M<R0,R1,R2,R3,R4,R5>
                       05 03B6 1319          RSB
```

```
                     03B7  1321          .SBTTL  DBG$CVT_MULP_R1
                     03B7  1322
                     03B7  1323 ;++
                     03B7  1324 ; FUNCTIONAL DESCRIPTION:
                     03B7  1325 ;       Multiply a packed decimal number.
                     03B7  1326 ;
                     03B7  1327 ;
                     03B7  1328 ; CALLING SEQUENCE:
                     03B7  1329 ;       DBG$CVT_MULP_R1 (mulrlen.rw, mulraddr.ab, muldlen.rw,
                     03B7  1330 ;                        muldaddr.ab, prodlen.rw, prodaddr.ab )
                     03B7  1331 ;       This is a JSB entry point.
                     03B7  1332 ;
                     03B7  1333 ;
                     03B7  1334 ; FORMAL PARAMETERS:
                     03B7  1335 ;       R0 --> mulrlen  R1 = mulraddr   R2 --> muldlen  R3 = muldaddr
                     03B7  1336 ;       R4 --> prodlen  R5 = prodaddr
                     03B7  1337 ;
                     03B7  1338 ;
                     03B7  1339 ; IMPLICIT INPUTS:
                     03B7  1340 ;       NONE
                     03B7  1341 ;
                     03B7  1342 ;
                     03B7  1343 ; IMPLICIT OUTPUTS:
                     03B7  1344 ;       NONE
                     03B7  1345 ;
                     03B7  1346 ;
                     03B7  1347 ; COMPLETION CODES:
                     03B7  1348 ;       NONE
                     03B7  1349 ;
                     03B7  1350 ; SIDE EFFECTS:
                     03B7  1351 ;       NONE
                     03B7  1352 ;
                     03B7  1353 ;--
                     03B7  1354 DBG$CVT_MULP_R1 ::
                 3F  BB  03B7  1355          PUSHR   #^M<R0,R1,R2,R3,R4,R5>
65   64   63   62   61  60  25  03B9  1356          MULP    (R0), (R1), (R2), (R3), (R4), (R5)
                 3F  BA  03C0  1357          POPR    #^M<R0,R1,R2,R3,R4,R5>
                     05  03C2  1358          RSB
```

```
                                        03C3  1360          .SBTTL  DBG$CVT_DIVP_R1
                                        03C3  1361
                                        03C3  1362 ;++
                                        03C3  1363 ; FUNCTIONAL DESCRIPTION:
                                        03C3  1364 ;       Divide a packed decimal number.
                                        03C3  1365 ;
                                        03C3  1366 ;
                                        03C3  1367 ; CALLING SEQUENCE:
                                        03C3  1368 ;       DBG$CVT_MULP_R1 (divrlen.rw, divraddr.ab, divdlen.rw,
                                        03C3  1369 ;                        divdaddr.ab, quolen.rw, quoaddr.ab )
                                        03C3  1370 ;       This is a JSB entry point.
                                        03C3  1371 ;
                                        03C3  1372 ;
                                        03C3  1373 ; FORMAL PARAMETERS:
                                        03C3  1374 ;       R0 --> divrlen  R1 = divraddr   R2 --> divdlen  R3 = divdaddr
                                        03C3  1375 ;       R4 --> quolen   R5 = quoaddr
                                        03C3  1376 ;
                                        03C3  1377 ;
                                        03C3  1378 ; IMPLICIT INPUTS:
                                        03C3  1379 ;       NONE
                                        03C3  1380 ;
                                        03C3  1381 ;
                                        03C3  1382 ; IMPLICIT OUTPUTS:
                                        03C3  1383 ;       NONE
                                        03C3  1384 ;
                                        03C3  1385 ;
                                        03C3  1386 ; COMPLETION CODES:
                                        03C3  1387 ;       NONE
                                        03C3  1388 ;
                                        03C3  1389 ; SIDE EFFECTS:
                                        03C3  1390 ;       NONE
                                        03C3  1391 ;
                                        03C3  1392 ;--
                                        03C3  1393 DBG$CVT_DIVP_R1 ::
                                3F  BB  03C3  1394          PUSHR   #^M<R0,R1,R2,R3,R4,R5>
65  64  63  62  61  60  27  03C5  1395          DIVP    (R0), (R1), (R2), (R3), (R4), (R5)
                                3F  BA  03CC  1396          POPR    #^M<R0,R1,R2,R3,R4,R5>
                                    05  03CE  1397          RSB
```

```
                                03CF  1399          .SBTTL  DBG$CVT_CMPH_R1
                                03CF  1400
                                03CF  1401 ;++
                                03CF  1402 ; FUNCTIONAL DESCRIPTION:
                                03CF  1403 ;       Compare two H floating values
                                03CF  1404 ;
                                03CF  1405 ;
                                03CF  1406 ; CALLING SEQUENCE:
                                03CF  1407 ;       STATUS = DBG$CVT_CMPH_R1 ( src1.rh.r, src2.rh.r )
                                03CF  1408 ;       This is a JSB entry point.
                                03CF  1409 ;
                                03CF  1410 ;
                                03CF  1411 ; FORMAL PARAMETERS:
                                03CF  1412 ;       R0 --> src1     R1 --> src2
                                03CF  1413 ;
                                03CF  1414 ;
                                03CF  1415 ; IMPLICIT INPUTS:
                                03CF  1416 ;       NONE
                                03CF  1417 ;
                                03CF  1418 ;
                                03CF  1419 ; IMPLICIT OUTPUTS:
                                03CF  1420 ;       NONE
                                03CF  1421 ;
                                03CF  1422 ;
                                03CF  1423 ; COMPLETION CODES:
                                03CF  1424 ;       R0 = -1 if src1 < src2
                                03CF  1425 ;       R0 = 0  if src1 = src2
                                03CF  1426 ;       R0 = 1  if src1 > src2
                                03CF  1427 ;
                                03CF  1428 ; SIDE EFFECTS:
                                03CF  1429 ;       OPCODE RESERVED TO DIGITAL error is possible, see ENVIRONMENT.
                                03CF  1430 ;
                                03CF  1431 ;--
                                03CF  1432 DBG$CVT_CMPH_R1 ::
          61    60 71FD         03CF  1433          CMPH    (R0), (R1)
                06   13         03D3  1434          BEQL    10$
                07   19         03D5  1435          BLSS    20$
          50    01   CE         03D7  1436          MNEGL   #1, R0                  ;src1 > src2
                     05         03DA  1437          RSB                             ;Finished
                                03DB  1438 10$:
                50   D4         03DB  1439          CLRL    R0                      ;They are equal
                     05         03DD  1440          RSB                             ;Finished
                                03DE  1441 20$:
50    FFFFFFFF 8F   D0         03DE  1442          MOVL    #-1, R0                 ;src1 < src2
                     05         03E5  1443          RSB                             ;Finished
                                03E6  1444
```

```
                03E6  1446 .END
```

```
D32                                 00000062 R      01
DBG$CVT_ASHP_R1                     000003AB RG     01
DBG$CVT_CMPH_R1                     000003CF RG     01
DBG$CVT_CVTDR_R1                    000001D8 RG     01
DBG$CVT_CVTGH_R1                    000002E4 RG     01
DBG$CVT_CVTHD_R1                    000002DA RG     01
DBG$CVT_CVTHF_R1                    000002D5 RG     01
DBG$CVT_CVTHG_R1                    000002DF RG     01
DBG$CVT_CVTLB_R1                    00000000 RG     01
DBG$CVT_CVTLH_R1                    00000008 RG     01
DBG$CVT_CVTLW_R1                    00000004 RG     01
DBG$CVT_CVTRDB_R1                   000001A8 RG     01
DBG$CVT_CVTRHL_R1                   000001DD RG     01
DBG$CVT_CVTRHO_R1                   0000027F RG     01
DBG$CVT_CVTRHQ_R1                   000001E2 RG     01
DBG$CVT_CVTROUB_R1                  00000072 RG     01
DBG$CVT_CVTROUF_R1                  00000021 RG     01
DBG$CVT_CVTROUG_R1                  000000C3 RG     01
DBG$CVT_CVTROUH_R1                  000000F0 RG     01
DBG$CVT_DIVD2_R1                    0000039D RG     01
DBG$CVT_DIVH2_R1                    000003A6 RG     01
DBG$CVT_DIVP_R1                     000003C3 RG     01
DBG$CVT_MULD2_R1                    00000399 RG     01
DBG$CVT_MULH2_R1                    000003A1 RG     01
DBG$CVT_MULP_R1                     000003B7 RG     01
DBG$CVT_SCALE_OU_DOWN_BY_10_R1      00000318 RG     01
DBG$CVT_SCALE_OU_DOWN_BY_2_R1       00000370 RG     01
DBG$CVT_SCALE_OU_UP_BY_10_R1        000002E9 RG     01
DBG$CVT_SCALE_OU_UP_BY_2_R1         00000341 RG     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$SIGNAL                          ******** X      01
O_LEN                             = 00000004
SHIFT_OU_LEFT                       0000018C R      01
SS$_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	000003E6	(998.)	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	23	00:00:00.10	00:00:00.92
Command processing	77	00:00:00.70	00:00:04.42
Pass 1	217	00:00:05.76	00:00:20.01
Symbol table sort	0	00:00:00.68	00:00:01.58
Pass 2	222	00:00:02.79	00:00:08.94
Symbol table output	6	00:00:00.06	00:00:00.20
Psect synopsis output	2	00:00:00.03	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	549	00:00:10.12	00:00:36.10

The working set limit was 1350 pages.
37171 bytes (73 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 445 non-local and 51 local symbols.
1446 source lines were read in Pass 1, producing 13 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

469 GETS were required to define 4 macros.

There were no errors, warnings or information messages.

MACRO/DISABLE:TBK/LIS=LIS$:DBGCVTMAC/OBJ=OBJ$:DBGCVTMAC MSRC$:DBGCVTMAC/UPDATE=(ENH$:DBGCVTMAC)

DBGCVTMAC
LIS

DBGDEFINE
LIS

DBGDPC
LIS