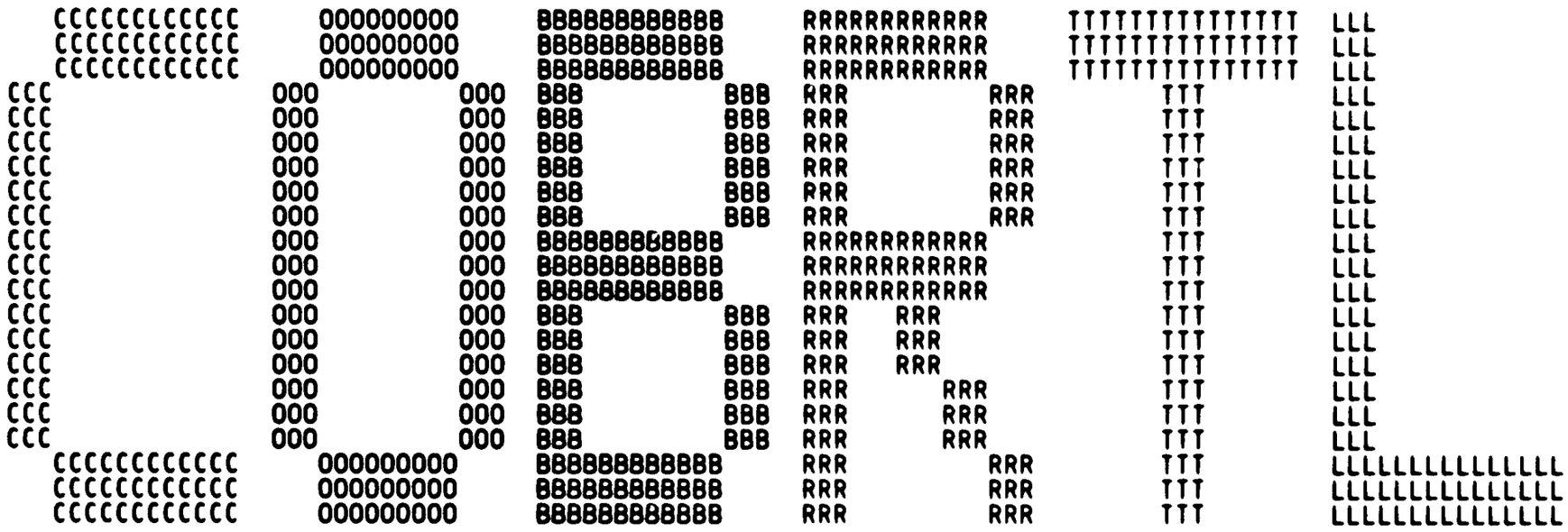
CCCCCCCCCCC 000000000 BBBBBBBBBBBB RRRRRRRRRRR TTTTTTTTTTTTTTT LLL
CCCCCCCCCCC 000000000 BBBBBBBBBBBB RRRRRRRRRRR TTTTTTTTTTTTTTT LLL
CCCCCCCCCCC 000000000 BBBBBBBBBBBB RRRRRRRRRRR TTTTTTTTTTTTTTT LLL
CCC 000 000 BBB BBB RRR RRR TTT LLL
CCC 000 000 BBB BBB RRR RRR TTT LLL
CCC 000 000 BBB BBB RRR RRR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

FILEID**COBEXPI

```
  CCCCCCCC    000000    BBBBBBBB    EEEEEEEEEE  XX      XX  PPPPPPPP      IIIIII
  CCCCCCCC    000000    BBBBBBBB    EEEEEEEEEE  XX      XX  PPPPPPPP      IIIIII
CC           00      00  BB      BB  EE          XX      XX  PP      PP      II
CC           00      00  BB      BB  EE          XX      XX  PP      PP      II
CC           00      00  BB      BB  EE            XX  XX    PP      PP      II
CC           00      00  BB      BB  EE            XX  XX    PP      PP      II
CC           00      00  BBBBBBBB    EEEEEEEE        XX      PPPPPPPP        II
CC           00      00  BBBBBBBB    EEEEEEEE        XX      PPPPPPPP        II
CC           00      00  BB      BB  EE            XX  XX    PP              II
CC           00      00  BB      BB  EE            XX  XX    PP              II
CC           00      00  BB      BB  EE          XX      XX  PP              II          ....
CC           00      00  BB      BB  EE          XX      XX  PP              II          ....
  CCCCCCCC    000000    BBBBBBBB    EEEEEEEEEE  XX      XX  PP          IIIIII          ....
  CCCCCCCC    000000    BBBBBBBB    EEEEEEEEEE  XX      XX  PP          IIIIII          ....

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

Table of contents

```
0000     1          .TITLE COB$EXPI         COBOL Intermediate Exponentiate
0000     2          .IDENT /1-012/          ; File: COBEXPI.MAR Edit:LGB1012
0000     3 ;
0000     4 ;*****************************************************************************
0000     5 ;*                                                                           *
0000     6 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     7 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     8 ;*  ALL RIGHTS RESERVED.                                                     *
0000     9 ;*                                                                           *
0000    10 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    11 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    12 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    13 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    14 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    15 ;*  TRANSFERRED.                                                             *
0000    16 ;*                                                                           *
0000    17 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    18 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    19 ;*  CORPORATION.                                                             *
0000    20 ;*                                                                           *
0000    21 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    22 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    23 ;*                                                                           *
0000    24 ;*                                                                           *
0000    25 ;*****************************************************************************
0000    26 ;
0000    27 ;
0000    28 ;       VERSION: 1
0000    29 ;
0000    30 ;       HISTORY:
0000    31 ;
0000    32 ;       AUTHOR:
0000    33 ;               Peter D Gilbert, 2-Aug-1979
0000    34 ;
0000    35 ;       MODIFIED BY:
0000    36 ;
0000    37 ;
```

```
0000     39          .SBTTL HISTORY          ; Detailed current edit history
0000     40 ;
0000     41 ; Edit history for Version 1 of COBINTARI.MAR
0000     42 ;
0000     43 ; 1-001 - Original, with input and output multiplexors lifted from COBINTARI.
0000     44 ;         PDG 2-Aug-1979
0000     45 ; 1-002 - Changed handling of negative bases.  PDG 22-Sep-1979.
0000     46 ; 1-003 - Make routine PIC.   RKR 24-SEPT-79
0000     47 ; 1-004 - Correct Undefined Exponentiation error code for LIB$SIGNAL
0000     48 ;         RKR 27-SEPT-79
0000     49 ; 1-005 - Remove additional NON-PIC instruction MOVAW 1[R0], R0
0000     50 ;         RKR 1-OCT-79
0000     51 ; 1-006 - Add additional entry point COB$EXPI_OSE, and minor code improvements
0000     52 ;         PDG 3-OCT-79
0000     53 ; 1-007 - Complete implementation of COB$EXPI_OSE.  MLJ 09-Oct-79
0000     54 ; 1-008 - Replaced OTS$_FATINTERR signals by COB$_INVARG.
0000     55 ;         Replaced all LIB$SIGNAL calls by LIB$STOP calls.
0000     56 ;         Cosmetic changes.  RKR 21-OCT-79
0000     57 ; 1-009 - Added checks for out-of-range CIT's.  RKR 30-OCT-79
0000     58 ; 1-010 - Fix check for exponent overflow and underflow in FINISH.
0000     59 ;         Guarantee that a fraction of zero has an exponent of zero.
0000     60 ;         RKR 23-APR-80
0000     61 ; 1-011 - Changed the branch to FINISH in routine COB$EXPI at label 1$:
0000     62 ;         to a RET instruction since FINISH expects the input argument
0000     63 ;         to be in the proper format, where in this case, the argument is
0000     64 ;         in error and therefore it was never put in the format that
0000     65 ;         FINISH expects it to be.  LB  15-APR-81
0000     66 ; 1-012 - Avoid undetected overflow of y in the exponentiate loop by adjusting
0000     67 ;         the scale of y.  Also, restore "begin-loop" and "end-loop" comments.
0000     68 ;         PDG 28-JUL-82
0000     69 ;         Prevent a negative base with zero exponent from getting the error
0000     70 ;         COB$_UNDEF_EXP.  It should get a result of 1.  A negative base
0000     71 ;         with an exponent less than zero will continue to correctly get the
0000     72 ;         COB$_UNDEF_EXP error.    LGB  17-AUG-82
0000     73 ;
```

```
                        .SBTTL DECLARATIONS

                        .DSABL  GBL

                ;
                ; INCLUDE FILES:
                ;

                        $DSCDEF
                        $INTDEF

                ;
                ; EXTERNAL SYMBOLS:
                ;
                        .EXTRN  COB$CVTWI_R8            ; Word to intermediate
                        .EXTRN  COB$CVTLI_R8            ; Longword to intermediate
                        .EXTRN  COB$CVTQI_R8            ; Quadword to intermediate
                        .EXTRN  COB$CVTFI_R7            ; Floating to intermediate
                        .EXTRN  COB$CVTDI_R7            ; Double to intermediate
                        .EXTRN  COB$CVTPI_R9            ; Packed to intermediate
                        .EXTRN  COB$CVTIW_R8            ; Intermediate to word
                        .EXTRN  COB$CVTIL_R8            ; Intermediate to longword
                        .EXTRN  COB$CVTIQ_R8            ; Intermediate to quadword
                        .EXTRN  COB$CVTIF_R7            ; Intermediate to floating
                        .EXTRN  COB$CVTID_R7            ; Intermediate to double
                        .EXTRN  COB$CVTIP_R9            ; Intermediate to packed
                        .EXTRN  COB$_INVARG             ; Signal -- Invalid arguments
                        .EXTRN  COB$_INTRESOPE          ; Intermediate reserved operand
                        .EXTRN  COB$_INTEXPUND          ; Intermediate underflow
                        .EXTRN  COB$_INTEXPOVE          ; Intermediate overflow
                        .EXTRN  COB$_UNDEF_EXP          ; Undefined exponentiation
                        .EXTRN  LIB$STOP                ; Signal msg and stop

                ;
                ; MACROS:
                ;
                        .MACRO  DCL  SYM,SIZ    ; To define offsets of stack temps
                SYM     =       OFFSET
                OFFSET  =       OFFSET + SIZ
                        .ENDM
00000000        OFFSET  =       0

                ;
                ; PSECT DECLARATIONS
                ;
00000000                .PSECT  _COB$CODE       PIC, SHR, LONG, EXE, NOWRT

                ;
                ; EQUATED SYMBOLS:
                ;
00000002        INT$P_I_FRACT=  2               ; Temporary until Packed supported in MDL
                                                ; Fraction field offset
                ;+
                ;
                ; OWN STORAGE:
                ;
                ;+
```

```
                                                0000   132
                                                0000   133 ;
                                                0000   134 ; log10( 10^k / (10^k-1) );  k=1,2,...,31 ;  ln( 1 + 10^(-k) )
                                                0000   135 ;
                                    FFFFFFF0    0000   136 TABLE1  = .-16
44 99 40 25 51 67 60 05 49 57 57 04             0000   137 .PACKED +045757490560675125409944193 4852  ; +010536051565782630122750098 0829
                                    2C 85 34 19 000C
44 97 65 84 00 45 02 54 80 64 43 00             0010   138 .PACKED +004364805402450084659744222 2456  ; +001005033585350144118354885 7556
                                    6C 45 22 22 001C
65 64 30 91 76 01 74 17 51 34 04 00             0020   139 .PACKED +000434511774017691306465600 6946  ; +000100050033358353350014298 2252
                                    6C 94 06 60 002C
56 45 38 10 75 80 19 16 43 43 00 00             0030   140 .PACKED +000043431619807510384556044 0232  ; +000010000500033358335333500 013
                                    2C 23 40 04 003C
52 93 37 01 39 53 66 29 34 04 00 00             0040   141 .PACKED +000004342966533901379352148 6461  ; +000001000005000033335833353 333
                                    1C 46 86 14 004C
44 75 63 50 90 69 94 42 43 00 00 00             0050   142 .PACKED +000000434294699050637544212 9173  ; +000000100000050000033333358 3333
                                    3C 17 29 21 005C
37 77 79 61 03 45 29 34 04 00 00 00             0060   143 .PACKED +000000043429450361797737046 2100  ; +000000010000000500000033333 358
                                    0C 10 62 04 006C
42 72 74 40 48 94 42 43 00 00 00 00             0070   144 .PACKED +000000004342944840747242516 4385  ; +000000001000000005000000033 3333
                                    5C 38 64 51 007C
99 03 12 82 44 29 34 04 00 00 00 00             0080   145 .PACKED +000000000434294482120399068 7473  ; +000000000100000000050000000 333
                                    3C 47 87 06 008C
96 24 19 48 94 42 43 00 00 00 00 00             0090   146 .PACKED +000000000043429448192496655 1746  ; +000000000010000000000500000 0000
                                    6C 74 51 65 009C
54 90 81 44 29 34 04 00 00 00 00 00             00A0   147 .PACKED +000000000004342944819054233 0006  ; +000000000001000000000005000 0000
                                    6C 00 30 23 00AC
03 19 48 94 42 43 00 00 00 00 00 00             00B0   148 .PACKED +000000000000434294481903468 9748  ; +000000000000100000000000050 0000
                                    8C 74 89 46 00BC
90 81 44 29 34 04 00 00 00 00 00 00             00C0   149 .PACKED +000000000000043429448190327 3542  ; +000000000000010000000000000 5000
                                    2C 54 73 32 00CC
19 48 94 42 43 00 00 00 00 00 00 00             00D0   150 .PACKED +000000000000004342944819032 5399  ; +000000000000001000000000000 0050
                                    9C 39 25 03 00DC
81 44 29 34 04 00 00 00 00 00 00 00             00E0   151 .PACKED +000000000000000434294481903 2518  ; +000000000000000100000000000 0000
                                    8C 51 32 90 00EC
48 94 42 43 00 00 00 00 00 00 00 00             00F0   152 .PACKED +000000000000000043429448190 3251  ; +000000000000000010000000000 0000
                                    1C 25 03 19 00FC
44 29 34 04 00 00 00 00 00 C0 00 00             0100   153 .PACKED +000000000000000004342944819 0325  ; +000000000000000001000000000 0000
                                    5C 32 90 81 010C
94 42 43 00 00 00 00 00 00 00 00 00             0110   154 .PACKED +000000000000000000434294481 9032  ; +000000000000000000100000000 0000
                                    2C 03 19 48 011C
29 34 04 00 00 00 00 00 00 00 00 00             0120   155 .PACKED +000000000000000000043429448 1903  ; +000000000000000000010000000 0000
                                    3C 90 81 44 012C
42 43 00 00 00 00 00 00 00 00 00 00             0130   156 .PACKED +000000000000000000004342944 8190  ; +000000000000000000001000000 0000
                                    0C 19 48 94 013C
34 04 00 00 00 00 00 00 00 00 00 00             0140   157 .PACKED +000000000000000000000434294 4819  ; +000000000000000000000100000 0000
                                    9C 81 44 29 014C
43 00 00 00 00 00 00 00 00 00 00 00             0150   158 .PACKED +000000000000000000000043429 4481  ; +000000000000000000000010000 0000
                                    1C 48 94 42 015C
04 00 00 00 00 00 00 00 00 00 00 00             0160   159 .PACKED +000000000000000000000004342 9448  ; +000000000000000000000001000 0000
                                    8C 44 29 34 016C
00 00 00 00 00 00 00 00 00 00 00 00             0170   160 .PACKED +000000000000000000000000434 2944  ; +000000000000000000000000100 0000
                                    4C 94 42 43 017C
00 00 00 00 00 00 00 00 00 00 00 00             0180   161 .PACKED +000000000000000000000000043 4294  ; +000000000000000000000000010 0000
                                    4C 29 34 04 018C
00 00 00 00 00 00 00 00 00 00 00 00             0190   162 .PACKED +000000000000000000000000004 3429  ; +000000000000000000000000001 0000
                                    9C 42 43 00 019C
```

```
00 00 00 00 00 00 00 00 00 00 00 00  01A0    163 .PACKED +0000000000000000000000000004342  ; +0000000000000000000000000001000
                        2C 34 04 00  01AC
00 00 00 00 00 00 00 00 00 00 00 00  01B0    164 .PACKED +0000000000000000000000000000434  ; +0000000000000000000000000000100
                        4C 43 00 00  01BC
00 00 00 00 00 00 00 00 00 00 00 00  01C0    165 .PACKED +0000000000000000000000000000043  ; +0000000000000000000000000000010
                        3C 04 00 00  01CC
00 00 00 00 00 00 00 00 00 00 00 00  01D0    166 .PACKED +0000000000000000000000000000004  ; +0000000000000000000000000000001
                        4C 00 00 00  01DC
00 00 00 00 00 00 00 00 00 00 00 00  01E0    167 .PACKED +0000000000000000000000000000000  ; +0000000000000000000000000000000
                        0C 00 00 00  01EC
                                     01F0    168
                                     01F0    169
                                     01F0    170 ;
                                     01F0    171 ; log10( 1 + 10^(-k) );  k=0,1,2,...,31   ;  ln( 1 + 10^(-k) )
                                     01F0    172 ;
                           000001F0  01F0    173 TABLE2  =.
38 37 21 95 11 98 63 56 99 29 10 30  01F0    174 .packed +3010299956639811952137388947245
                        5C 24 47 89  01FC
99 01 75 40 50 22 58 51 68 92 13 04  0200    175 .PACKED +0413926851582250407501999712422  ; +0095310179804324860043952123279
                        2C 42 12 97  020C
88 51 27 74 25 64 82 37 37 21 43 00  0210    176 .PACKED +0043213737826425742751881782228  ; +0009950330853168082848215357544
                        8C 22 82 17  021C
21 89 66 40 86 31 79 74 07 34 04 00  0220    177 .PACKED +0004340774793186406689213877777  ; +0000999500333083533166809398920
                        7C 77 77 38  022C
13 73 63 69 26 86 76 72 42 43 00 00  0230    178 .PACKED +0000434272768626696373135275851  ; +0000099995000333308335333166681
                        1C 85 75 52  023C
55 68 18 53 44 10 23 29 34 04 00 00  0240    179 .PACKED +0000043429231044531868554934715  ; +0000009999950000333330833353333
                        5C 71 34 49  024C
40 56 15 56 47 26 94 42 43 00 00 00  0250    180 .PACKED +0000004342942647561556407439424  ; +0000000999999500000333333083333
                        4C 42 39 74  025C
18 29 85 18 60 44 29 34 04 00 00 00  0260    181 .PACKED +0000000434294460188529180136700  ; +0000000099999995000000333333308
                        0C 70 36 01  026C
94 77 31 97 47 94 42 43 00 00 00 00  0270    182 .PACKED +0000000043429447973177943261133  ; +0000000009999999950000000333333
                        3C 13 61 32  027C
04 61 68 81 44 29 34 04 00 00 00 00  0280    183 .PACKED +0000000004342944816861045868441  ; +0000000000999999999500000000333
                        1C 44 68 58  028C
53 81 18 48 94 42 43 00 00 00 00 00  0290    184 .PACKED +0000000000434294481881537103555  ; +0000000000099999999995000000000
                        5C 55 03 71  029C
10 90 81 44 29 34 04 00 00 00 00 00  02A0    185 .PACKED +0000000000043429448190108035524  ; +0000000000009999999999950000000
                        4C 52 35 80  02AC
03 19 48 94 42 43 00 00 00 00 00 00  02B0    186 .PACKED +0000000000004342944819030346804  ; +0000000000000999999999999500000
                        4C 80 46 03  02BC
90 81 44 29 34 04 00 00 00 00 00 00  02C0    187 .PACKED +0000000000004342944819032301l2  ; +0000000000000099999999999995000
                        2C 11 30 32  02CC
19 48 94 42 43 00 00 00 00 00 00 00  02D0    188 .PACKED +0000000000000043429448190324965  ; +0000000000000009999999999999950
                        5C 96 24 03  02DC
81 44 29 34 04 00 00 00 00 00 00 00  02E0    189 .PACKED +0000000000000004342944819032518  , +0000000000000001000000000000000
                        8C 51 32 90  02EC
48 94 42 43 00 00 00 00 00 00 00 00  02F0    190 .PACKED +0000000000000000434294481903251  ; +0000000000000000100000000000000
                        1C 25 03 19  02FC
44 29 34 04 00 00 00 00 00 00 00 00  0300    191 .PACKED +0000000000000000043429448190325  ; +0000000000000000010000000000000
                        5C 32 90 81  030C
94 42 43 00 00 00 00 00 00 00 00 00  0310    192 .PACKED +0000000000000000004342944819032  ; +0000000000000000001000000000000
                        2C 03 19 48  031C
29 34 04 00 00 00 00 00 00 00 00 00  0320    193 .PACKED +0000000000000000000434294481903  ; +0000000000000000000100000000000
                        3C 90 81 44  032C
42 43 00 00 00 00 00 00 00 00 00 00  0330    194 .PACKED +0000000000000000000043429448190  ; +0000000000000000000010000000000
```

```
                                0C 19 48 94   033C
34 04 00 00 00 00 00 00 00 00 00 00   0340   195 .PACKED +0000000000000000000004342944819  ; +0000000000000000000001000000000
                                9C 81 44 29   034C
43 00 00 00 00 00 00 00 00 00 00 00   0350   196 .PACKED +0000000000000000000000434294481  ; +0000000000000000000000100000000
                                1C 48 94 42   035C
04 00 00 00 00 00 00 00 00 00 00 00   0360   197 .PACKED +0000000000000000000000043429448  ; +0000000000000000000000010000000
                                8C 44 29 34   036C
00 00 00 00 00 00 00 00 00 00 00 00   0370   198 .PACKED +0000000000000000000000004342944  ; +0000000000000000000000001000000
                                4C 94 42 43   037C
00 00 00 00 00 00 00 00 00 00 00 00   0380   199 .PACKED +0000000000000000000000000434294  ; +0000000000000000000000000100000
                                4C 29 34 04   038C
00 00 00 00 00 00 00 00 00 00 00 00   0390   200 .PACKED +0000000000000000000000000043429  ; +0000000000000000000000000010000
                                9C 42 43 00   039C
00 00 00 00 00 00 00 00 00 00 00 00   03A0   201 .PACKED +0000000000000000000000000004342  ; +0000000000000000000000000001000
                                2C 34 04 00   03AC
00 00 00 00 00 00 00 00 00 00 00 00   03B0   202 .PACKED +0000000000000000000000000000434  ; +0000000000000000000000000000100
                                4C 43 00 00   03BC
00 00 00 00 00 00 00 00 00 00 00 00   03C0   203 .PACKED +0000000000000000000000000000043  ; +0000000000000000000000000000010
                                3C 04 00 00   03CC
00 00 00 00 00 00 00 00 00 00 00 00   03D0   204 .PACKED +0000000000000000000000000000004  ; +0000000000000000000000000000001
                                4C 00 00 00   03DC
00 00 00 00 00 00 00 00 00 00 00 00   03E0   205 .PACKED +0000000000000000000000000000000  ; +0000000000000000000000000000000
                                0C 00 00 00   03EC
                                              03F0   206
                                           0C 03F0   207 P0:     .packed +0      ; Temp'ry 'til ass'bler ass'bles sh'rt p'cked const's.
                                           1C 03F1   208 P1:     .packed +1
99 99 99 99 99 99 99 99 99 99 99 99   03F2   209 NINES:  .packed +999999999999999999999999999999999        ; 31 nines
                                9C 99 99 99   03FE
                                              0402   210
```

```
                        0402   212          .SBTTL  MULBIG          Packed Multiply of Big numbers
                        0402   213
                        0402   214 ;++
                        0402   215 ;
                        0402   216 ; FUNCTIONAL DESCRIPTION:
                        0402   217 ;
                        0402   218 ;       Perform a multiply of two 31-digit packed numbers, producing a
                        0402   219 ;       63-digit 'packed' result.
                        0402   220 ;
                        0402   221 ; CALLING SEQUENCE:
                        0402   222 ;
                        0402   223 ;       MULBIG (MULRADDR.ap, MULDADDR.ap, PRODADDR.ap)
                        0402   224 ;
                        0402   225 ; INPUT PARAMETERS:
                        0402   226 ;
                        0402   227 ;       MULRADDR.ap             Address of Multiplier
                        0402   228 ;       MULDADDR.ap             Address of Multiplicand
                        0402   229 ;       PRODADDR.ap             Address of Product (32 bytes)
                        0402   230 ;
                        0402   231 ; IMPLICIT INPUTS:
                        0402   232 ;
                        0402   233 ;       The multiplier and multiplicand are modi'ied during the multiply,
                        0402   234 ;       but are left unchanged at exit.  Overlapping operands will produce
                        0402   235 ;       incorrect results.
                        0402   236 ;
                        0402   237 ; IMPLICIT OUTPUTS:
                        0402   238 ;
                        0402   239 ;       NONE
                        0402   240 ;
                        0402   241 ;--
                        0402   242
                        0402   243 ;       .ENTRY  MULBIG, ^M<R2,R3,R4,R5,R8>      ; Don't want this to be global
                   013C 0402   244 MULBIG: .WORD           ^M<R2,R3,R4,R5,R8>
                        0404   245
                   00000000 0404   246 A$      = 0
                   00000012 0404   247 B$      = 18 + A$
                   00000024 0404   248 C$      = 18 + B$
                   00000030 0404   249 D$      = 12 + C$
                   0000003C 0404   250 E$      = 12 + D$
                        0404   251
                5E   3C   C2 0404   252         SUBL2   #E$,SP
                        0407   253
             58   0C AC   D0 0407   254         MOVL    12(AP),R8
                        040B   255
02 AE    04 BC   1F   34 040B   256         MOVP    #31,@4(AP),   2+A$(SP)
   6E    02 AE   05   28 0411   257         MOVC3   #5,2+0+A$(SP),0+A$(SP)
06 AE    07 AE   05   28 0416   258         MOVC3   #5,2+5+A$(SP),6+A$(SP)
50   11 AE   F0 8F   8B 041C   259         BICB3   #^XF0,5+12+A$(SP),R0
         0B AE   50   90 0422   260         MOVB    R0,   5+ 6+A$(SP)
         05 AE   50   90 0426   261         MOVB    R0,   5+ 0+A$(SP)
                        042A   262
14 AE    08 BC   1F   34 042A   263         MOVP    #31,@8(AP),   2+B$(SP)
12 AE    14 AE   05   28 0430   264         MOVC3   #5,2+0+B$(SP),0+B$(SP)
18 AE    19 AE   05   28 0436   265         MOVC3   #5,2+5+B$(SP),6+B$(SP)
50   23 AE   F0 8F   8B 043C   266         BICB3   #^XF0,5+12+B$(SP),R0
         1D AE   50   90 0442   267         MOVB    R0,   5+ 6+B$(SP)
         17 AE   50   90 0446   268         MOVB    R0,   5+ 0+B$(SP)
```

```
                                                044A   269
                                    68   7C     044A   270          CLRQ    0(R8)
                                 08 A8   7C     044C   271          CLRQ    8(R8)
  1F    1E AE    0B    0C AE     0B   25     044F   272          MULP    #11,12+A$(SP),#11,12+B$(SP),#31,16(R8)  ; A0xB0
                                    10 A8       0457
            50    23 AE    11 AE   8D     0459   273          XORB3     5+12+A$(SP),  5+12+B$(SP), R0
                  1F A8    50      0C     045F   274          BISB3   #^X0C,R0,31(R8)
                                                0464   275
  17    18 AE    0B    0C AE     0B   25     0464   276          MULP    #11,12+A$(SP),#11, 6+B$(SP),#23,C$(SP)  ; A0xB1
                                    24 AE       046C
  17    1E AE    0B    06 AE     0B   25     046E   277          MULP    #11, 6+A$(SP),#11,12+B$(SP),#23,D$(SP)  ; A1xB0
                                    30 AE       0476
       24 AE    17    30 AE     17   20     0478   278          ADDP4   #23,   D$(SP),#23,   C$(SP)
                                 54    1A A8   90     047F   279          MOVB                    11+15(R8), R4
         1A A8    1F A8     F0 8F     8B     0483   280          BICB3   #^XF0, 31(R8),  11+15(R8)
       0F A8    17   24 AE     17   20     048A   281          ADDP4   #23,   C$(SP),#23, 15(R8)
                     1A A8     54   90     0491   282          MOVB    R4,            11+15(R8)
                                                0495   283
  17    12 AE    0B    0C AE     0B   25     0495   284          MULP    #11,12+A$(SP),#11, 0+B$(SP),#23,C$(SP)  ; A0xB2
                                    24 AE       049D
  17    18 AE    0B    06 AE     0B   25     049F   285          MULP    #11, 6+A$(SP),#11, 6+B$(SP),#23,D$(SP)  ; A1xB1
                                    30 AE       04A7
       24 AE    17    30 AE     17   20     04A9   286          ADDP4   #23,   D$(SP),#23,   C$(SP)
     17    1E AE    0B    6E     0B   25     04B0   287          MULP    #11, 0+A$(SP),#11,12+B$(SP),#23,D$(SP)  ; A2xB0
                                    30 AE       04B7
       24 AE    17    30 AE     17   20     04B9   288          ADDP4   #23,   D$(SP),#23,   C$(SP)
                                 54    15 A8   90     04C0   289          MOVB                    11+10(R8), R4
         15 A8    1F A8     F0 8F     8B     04C4   290          BICB3   #^XF0, 31(R8),  11+10(R8)
       0A A8    17   24 AE     17   20     04CB   291          ADDP4   #23,   C$(SP),#23, 10(R8)
                     15 A8     54   90     04D2   292          MOVB    R4,            11+10(R8)
                                                04D6   293
  17    12 AE    0B    06 AE     0B   25     04D6   294          MULP    #11, 6+A$(SP),#11, 0+B$(SP),#23,C$(SP)  ; A1xB2
                                    24 AE       04DE
     17    18 AE    0B    6E     0B   25     04E0   295          MULP    #11, 0+A$(SP),#11, 6+B$(SP),#23,D$(SP)  ; A2xB1
                                    30 AE       04E7
       24 AE    17    30 AE     17   20     04E9   296          ADDP4   #23,   D$(SP),#23,   C$(SP)
                                 54    10 A8   90     04F0   297          MOVB                    11+ 5(R8), R4
         10 A8    1F A8     F0 8F     8B     04F4   298          BICB3   #^XF0, 31(R8),  11+ 5(R8)
       05 A8    17   24 AE     17   20     04FB   299          ADDP4   #23,   C$(SP),#23,  5(R8)
                     10 A8     54   90     0502   300          MOVB    R4,            11+ 5(R8)
                                                0506   301
     17    12 AE    0B    6E     0B   25     0506   302          MULP    #11, 0+A$(SP),#11, 0+B$(SP),#23,C$(SP)  ; A2xB2
                                    24 AE       050D
                                 54    0B A8   90     050F   303          MOVB                    11+ 0(R8), R4
         0B A8    1F A8     F0 8F     8B     0513   304          BICB3   #^XF0, 31(R8),  11+ 0(R8)
        68    17   24 AE     17   20     051A   305          ADDP4   #23,   C$(SP),#23,  0(R8)
                     0B A8     54   90     0520   306          MOVB    R4,            11+ 0(R8)
                                                0524   307
                                    04      0524   308          RET
                                                0525   309
```

```
0525   311          .SBTTL  COB$EXPI        Exponentiate intermediate temporary
0525   312
0525   313 ;++
0525   314 ;
0525   315 ; FUNCTIONAL DESCRIPTION:
0525   316 ;
0525   317 ;       Accept any two supported data types as input, convert them to
0525   318 ;       Intermediate, exponentiate them, convert the Intermediate result
0525   319 ;       to the data type of the output argument, and return.
0525   320 ;
0525   321 ;       1.  If routine is confronted with unknown data type it
0525   322 ; SIGNALSTOPs COB$_INVARG.
0525   323 ;
0525   324 ;       2.  If presented with an input CIT which has an overflowed or
0525   325 ; underflowed exponent field it SIGNALSTOPs COB$_INTRESOPE.
0525   326 ;
0525   327 ;       3.  If entered at COB$EXPI_OSE  (on size error) and
0525   328 ; exponentiation can't be done (e.g. exp < 0), returns 0.
0525   329 ;
0525   330 ;       4.  If entered at COB$EXPI and exponentiation can't be done
0525   331 ; it SIGNALSTOPs COB$_UNDEXP.
0525   332 ;
0525   333 ;       5.  If exponentiation is performed and
0525   334 ;                       If resulting CIT has overflowed exponent field,
0525   335 ;                       SIGNALSTOP COB$_INTEXPOVE.
0525   336 ;
0525   337 ;                       If resulting CIT has underflowed exponent field,
0525   338 ;                       SIGNALSTOP COB$_INTEXPUND.
0525   339 ; CALLING SEQUENCE:
0525   340 ;
0525   341 ;       COB$EXPI        (BASE.rx.dx, EXPONENT.rx.dx, POWER.wx.dx)
0525   342 ;       COB$EXPI_OSE    (BASE.rx.dx, EXPONENT.rx.dx, POWER.wx.dx)
0525   343 ;
0525   344 ; INPUT PARAMETERS:
0525   345 ;
0525   346 ;       BASE.rx.dx              The operand to the left of the operator
0525   347 ;       EXPONENT.rx.dx          The operand to the right of the operator
0525   348 ;
0525   349 ; IMPLICIT INPUTS:
0525   350 ;
0525   351 ;       NONE
0525   352 ;
0525   353 ; OUTPUT PARAMETERS:
0525   354 ;
0525   355 ;       POWER.wx.dx             The power BASE ** EXPONENT
0525   356 ;
0525   357 ; IMPLICIT OUTPUTS:
0525   358 ;
0525   359 ;       Different error handling for different entry points:
0525   360 ;       COB$EXPI -      Call LIB$STOP for bad exponentiation.
0525   361 ;       COB$EXPI_OSE -  Return R0 = 1 or 0 for success or failure.
0525   362 ;
0525   363 ; FUNCTION VALUE:
0525   364 ;
0525   365 ;       COB$EXPI_OSE -  1 or 0, depending on success or failure.
0525   366 ;
0525   367 ; SIDE EFFECTS:
```

```
0525   368 ;
0525   369 ;       NONE
0525   370 ;
0525   371 ; QUOTES:
0525   372 ;
0525   373 ;       "The invisible are insane."
0525   374 ;                       - English translation of a Chinese translation of
0525   375 ;                               an English proverb.
0525   376 ;
0525   377 ;       "The Stone the Builders Rejected".
0525   378 ;                       - Inscription on Jack London's gravestone.
0525   379 ;--
0525   380
0525   381         DCL     base_it,INT$K_I_LEN     ; intermediate temp
0525   382         DCL     exp_it, INT$K_I_LEN     ; intermediate temp
0525   383         DCL     res_it, INT$K_I_LEN     ; intermediate temp
0525   384         DCL     x,      16              ; for 31 digits
0525   385         DCL     y,      16              ; for 31 digits
0525   386         DCL     z,      32              ; for 63 digits
0525   387         DCL     res_sign,1              ; sign of result
0525   388         DCL     ose,    1               ; remember entry point
```

```
                        0525   390 ERR_0:                                  ; Base = 0
                  12 37 0525   391          CMPP4   #INT$K_I_FRACT_D,-
                  0E AE 0527   392                  INT$P_I_FRACT+exp_it(SP),-
              FEC3 CF   0529   393                  #1,P0
                    06  15 052D   394          BLEQ    ERR_BAD                 ; If exponent > 0 return 0.0 (= base)
                 50 01  D0 052F   395          MOVL    #1,R0
                    0235 31 0532   396          BRW     FINISH
                        0535   397 ERR_BAD:
                 0D 65 AE  E8 0535   398          BLBS    ose(SP),1$              ; Br if to return status
              00000000'8F  DD 0539   399          PUSHL   #COB$_UNDEF_EXP         ; Undefined exponentiation
         00000000'GF   01  FB 053F   400          CALLS   #1,G^LIB$STOP           ; Signal it and quit
                    50  D4 0546   401 1$:       CLRL    R0                      ; Error
                    04  04 0548   402          RET
                        0549   403
                    03FC 0549   404          .ENTRY  COB$EXPI_OSE,^M<R2,R3,R4,R5,R6,R7,R8,R9>
                 50 01  D0 054B   405          MOVL    #1,R0                   ; Remember flavor of call
                    04  11 054E   406          BRB     EXP_J
                    03FC 0550   407          .ENTRY  COB$EXPI,^M<R2,R3,R4,R5,R6,R7,R8,R9>
                        0552   408          ;
                        0552   409          ; Convert to intermediate
                        0552   410          ;
                    50  D4 0552   411          CLRL    R0
         5E 00000066 8F  C2 0554   412 EXP_J:  SUBL2   #offset,SP
               65 AE 50  90 055B   413          MOVB    R0,ose(SP)
               50 04 AC  D0 055F   414          MOVL    4(AP),R0
                  51 6E  9E 0563   415          MOVAB   base_it(SP),R1
                     0336 30 0566   416          BSBW    CONVERT
               50 08 AC  D0 0569   417          MOVL    8(AP),R0
               51 0C AE  9E 056D   418          MOVAB   exp_it(SP),R1
                     032B 30 0571   419          BSBW    CONVERT
                        0574   420
                        0574   421          ;
                        0574   422          ; Compute the log base 10 of the base
                        0574   423          ;
                        02 AE  95 0574   424          TSTB    base_it+INT$P_I_FRACT(SP)       ; See if base is zero
                           AC  13 0577   425          BEQL    ERR_0                   ; What to do? ####
                        0579   426          ;
                        0579   427          ; Determine the correct sign of the result
                        0579   428          ;
                     64 AE  0C  90 0579   429          MOVB    #^X0C,res_sign(SP)      ; Assume positive
                  24 AE  0B AE  10 057D   430          BISB3   #^X10,<INT$K_I_FRACT_D/2>+INT$P_I_FRACT+base_it(SP),x(SP)
               FE66 CF   24 AE  01 0583   431          CMPP3   #1,x(SP),P0
                           23  18 058A   432          BGEQ    1314$                   ; It is positive
                     54 0C AE  32 058C   433          CVTWL   INT$W_I_EXP+exp_it(SP),R4       ; If exp <= 0
                        0590   434          ;
                        0590   435          ; BLEQ instruction changed to the BLSS instruction below,
                        0590   436          ; see edit 1-012
                        0590   437          ;
                           A3  19 0590   438          BLSS    ERR_BAD                 ; then exponent isn't integral
                     12 54  F8 0592   439          ASHP    R4,#INT$K_I_FRACT_D,-           ; Look at the fractional part
            24 AE    12 00 0E AE       0595   440                  INT$P_I_FRACT+exp_it(SP),#0,#INT$K_I_FRACT_D,x(SP)
                           98  12 059B   441          BNEQ    ERR_BAD                 ; If non-zero, then bad
                     54 12  A2 059D   442          SUBW2   #INT$K_I_FRACT_D,R4             ; Look at the integer part
         63 01 00 61 12 54  F8 05A0   443          ASHP    R4,#INT$K_I_FRACT_D,(R1),#0,#1,(R3)
                  04 63 04  E1 05A7   444          BBC     #4,(R3),1314$           ; If even, result is positive
                  64 AE 0D  90 05AB   445          MOVB    #^X0D,res_sign(SP)      ; Else result is negative
                        05AF   446 ;
```

```
                                        05AF    447 ;       begin log loop
                                        05AF    448 ;
                                  0C    F8  05AF    449 1314$:  ASHP    #<30-INT$K_I_FRACT_D>,-
                               02 AE    12  05B1    450                 #INT$K_I_FRACT_D,base_it+INT$P_I_FRACT(SP),-
                            24 AE   1E  00  05B4    451                 #0,#30,x(SP)                        ; set x
                               33 AE    0F  88  05B8    452         BISB2   #^X0F,15+x(SP)                      ; make it positive
   1D   00   63   1E   FF   8F  F8  05BC    453         ASHP    #-1,#30,(R3),#0,#29,z(SP)           ; z <- x/10
                               44 AE        05C3
1F   00   FE25 CF   01   1E  F8  05C5    454         ASHP    #<31-1>,#1,P0,#0,#31,y(SP)          ; y <- 0
                               34 AE        05CD
                               56   01  CE  05CF    455         MNEGL   #1,R6                               ; k <- 1
                            01   24 AE  91  05D2    456 2$:     CMPB    x(SP),#1                            ; x = 1 ?
                                     12  14  05D6    457         BGTR    3$                                  ; br if x > 1
  FE10 CF   01   25 AE   1D  37  05D8    458         CMPP4   #29,1+x(SP),#1,P0
                                     38  1s  05E0    459         BEQL    10$                                 ; br if log calculation is done
         44 AE   25 AE   1D  35  05E2    460         CMPP3   #29,1+x(SP),z(SP)                   ; x-z >= 1 ?
                                     23  19  05E8    461         BLSS    4$                                  ; br if not
   24 AE   1F   44 AE   1D  22  05EA    462 3$:     SUBP4   #29,z(SP),#31,x(SP)                 ; x <- x-z
  1D   00   24 AE   1E   56  F8  05F1    463         ASHP    R6,#30,x(SP),#0,#29,z(SP)           ; z <- x / 10^k
                               44 AE        05F8
      50   56   FFFFFFF0 8F  C5  05FA    464         MULL3   #-16,R6,R0                          ; k * 16
34 AE   1F   F9E8 CF40   1F  20  0602    465         ADDP4   #31,Table1[R0],#31,y(SP)            ; y <- y + log( 10^k/(10^k-1) )
                                     C5  11  060B    466         BRB     2$
                                     56  D7  060D    467 4$:     DECL    R6                                  ; k <- k+1
  1D   00   24 AE   1E   56  F8  060F    468         ASHP    R6,#30,x(SP),#0,#29,z(SP)           ; z <- z / 10
                               44 AE        0616
                                     B8  12  0618    469         BNEQ    2$                                  ; finished if k gets very large
                                        061A    470 10$:
                                        061A    471 ;
                                        061A    472 ;       end log loop
                                        061A    473 ;
                                        061A    474 ;       00...0+ <= y <= 99...9+
                                        061A    475
                                        061A    476 ;       E_d is the number of extra digits we need for the exponent,
                                        061A    477 ;       rounded up to the nearest multiple of 2.
                                        061A    478
                                00000002    061A    479 E_d     =       2
                                        061A    480         .IIF    GE,INT$K_I_EXP_HI-100,          E_d = 4
                                        061A    481         .IIF    LT,INT$K_I_EXP_LO+100,          E_d = 4
                                        061A    482         .IIF    GE,INT$K_I_EXP_HI-10000,        E_d = 6
                                        061A    483         .IIF    LT,INT$K_I_EXP_LO+10000,        E_d = 6
                                        061A    484
                                        061A    485 ;       00...0+ <= y <  99...9+
                                        061A    486
                                        061A    487 ;
                                        061A    488 ;       begin multiply by exponent
                                        061A    489 ;
34 AE   1F   FDD3 CF   1F  22  061A    490         SUBP4   #31,NINES,#31,y(SP);                ; Subtract one (essentially)
                               50   6E  32  0622    491         CVTWL   INT$W_I_EXP+base_it(SP),R0          ; need to add this on
            44 AE   03   50   F9  0625    492         CVTLP   R0,#E_d+1,z(SP)
1F   00   44 AE   03   1D   F8  062A    493         ASHP    #31-E_d,#E_d+1,z(SP),#0,#31,x(SP)
                                  24 AE     0631
                               42 AE   0F  8A  0633    494         BICB2   #^X0F,15-<E_d/2>+y(SP)              ; Move sign closer in y
                               42 AE   0D  88  0637    495         BISB2   #^X0D,15-<E_d/2>+y(SP)              ; Remember it's negative now
  24 AE   1F   34 AE   1D  20  063B    496         ADDP4   #31-E_d,y(SP),#31,x(SP)             ; x <- y + exponent - 1
                                        0642    497
                                        0642    498
```

```
                 0D   F8  0642   499          ASHP    #31-INT$K_I_FRACT_D,-
      0E AE      12       0644   500                  #INT$K_I_FRACT_D,INT$P_I_FRACT+exp_it(SP),-
34 AE    1F     00       0647   501                  #0,#31,y(SP)
           44 AE   9F   064B   502          PUSHAB  z+0(SP)                         ; z <- y * x
           38 AE   9F   064E   503          PUSHAB  y+4(SP)                         ; (this is why z needs 63 digs)
           2C AE   9F   0651   504          PUSHAB  x+8(SP)
   FDA9 CF    03   FB  0654   505          CALLS   #3,MULBIG
                       0659   506
                       0659   507 ;        -099.9999  < z <  +099.9999
                       0659   508 ;
                       0659   509 ;        end multiply by exponent
                       0659   510 ;
```

```
                                0659   512 ;
                                0659   513 ;   We want to grab the integer and fractional parts of this product.
                                0659   514 ;   Unfortunately, the VAX does not (yet) support 63 digit packed numbers.
                                0659   515 ;
                                0659   516
            44 AE    1F    00   3B  0659   517          SKPC    #0,#31,z(SP)                    ; Skip leading zeroes
                     54    50   D0  065E   518          movl    R0,R4                           ; Save # of bytes remaining
                     03    50   D1  0661   519          cmpl    R0,#3                           ; Do we have at least 7 digits?
                           16   14  0664   520          bgtr    1001$                           ; Branch if so
               50    01 A040   9E  0666   521          movab   1(R0)[R0],R0                    ; # of digits in product
               52    07    50   C3  066B   522          subl3   R0,#7,R2                        ; Calculate shift amount
24 AE   07   00   61    50    52   F8  066F   523          ashp    R2,R0,(R1),#0,#7,x(SP)          ; Shift exponent into x
                           19   11  0677   524          brb     1002$                           ; Merge with other case
                        FEB9    31  0679   525 9191$:    BRW     ERR_BAD                         ; Branch point
             28 AE    03 A1   90  067C   526 1001$:    movb    3(R1),4+x(SP)                   ; Save these digs a moment
      03 A1  63 AE    F0 8F   8B  0681   527          bicb3   #^XF0,31+z(SP),3(R1)            ; Put in the correct sign
          24 AE    61    07   34  0688   528          movp    #7,(R1),x(SP)                   ; The exponent's hidden in here
            03 A1    28 AE   90  068D   529          movb    4+x(SP),3(R1)                   ; Restore digits
                                0692   530 1002$:
               56    0C AE   32  0692   531          CVTWL   INT$W_I_EXP+exp_it(SP),R6       ; Get the exponent's exponent
            50    BE A644   3E  0696   532          MOVAW   -68+E_d(R6)[R4],R0              ; Calculate shift amount (!)
   07   00   24 AE    07    50  F8  069B   533          ASHP    R0,#7,x(SP),#0,#7,4+x(SP)
                     28 AE       06A2
                           D3   1D  06A4   534          BVS     9191$                           ; Exponent overflow ####
            51    28 AE    07   36  06A6   535 1004$:    CVTPL   #7,4+x(SP),R1
                  18 AE    51   F7  06AB   536          CVTLW   R1,INT$W_I_EXP+res_it(SP)
                           C8   1D  06AF   537          BVS     9191$
                                06B1   538 ;
                                06B1   539 ;       The decimal point is E_d+1+R6 places to the right of z
                                06B1   540 ;
                           54   D4  06B1   541          CLRL    R4
                  57    03 A6   9E  06B3   542          MOVAB   E_d+1(R6),R7
                           0D   15  06B7   543          BLEQ    1010$               ; R4 is correct offset from z(SP)
            54    57    FF   8F   78  06B9   544          ASHL    #-1,R7,R4
                     10    54   D1  06BE   545          CMPL    R4,#16
                           03   15  06C1   546          BLEQ    1010$               ; R4 is correct offset from z(SP)
                        54   10   D0  06C3   547          MOVL    #16,R4              ; R4 is correct offset from z(SP)
                                06C6   548 1010$:
                                06C6   549 ;
                                06C6   550 ;       Shift by  E_d + 1 + R6 - 2*R4   = R7 - 2*R4
                                06C6   551 ;
      50    63 AE    F0 8F   8B  06C6   552          BICB3   #^XF0,31+z(SP),R0               ; Put in a correct sign
            53 AE44   0F   8A  06CC   553          BICB2   #^X0F,15+z(SP)[R4]
            53 AE44   50   88  06D1   554          BISB2   R0,15+z(SP)[R4]
 50   57    54   FFFFFFFE 8F   7A  06D6   555          EMUL    #-2,R4,R7,R0                    ; R7 - 2 * R4
1F   00   44 AE44   1F    50  F8  06DF   556          ASHP    R0,#31,z(SP)[R4],#0,#31,x(SP)   ; Shift into x
                     24 AE       06E7
                           13   18  06E9   557          BGEQ    1011$                           ; >= 0 ?  D'lovely.
  24 AE   1F   FD02 CF    1F   20  06EB   558          ADDP4   #31,NINES,#31,x(SP)             ; Add 1.0
  24 AE   1F   FCF9 CF    01   20  06F3   559          ADDP4   #1,P1,#31,x(SP)
                  18 AE    B7   06FB   560          DECW    INT$W_I_EXP+res_it(SP)          ; Decrease result's exponent
                                06FE   561 1011$:
                                06FE   562
```

```
                                                06FE   564 ;
                                                06FE   565 ;       begin exponentiate loop
                                                06FE   566 ;
                                                06FE   567 ;       +00000  <= x <=  +99999
                                                06FE   568
1F   00   FCED CF   01   1D   F8                06FE   569         ASHP    #29,#1,P1,#0,#31,y(SP)          ; y <- 1
                         34 AE                  0706
                            56   D4             0708   570         CLRL    R6                              ; k <- 0
     54   56   FFFFFFF0 8F   C5                 070A   571 11$:    MULL3   #-16,R6,R4
    24 AE    FAD8 CF44   1F   35                 0712   572         CMPP3   #31,table2[R4],x(SP)            ; log(1+10^(-k)) > x ?
                         1D   14                 071A   573         BGTR    12$
  1F   00   34 AE   1F   56   F8                 071C   574         ASHP    R6,#31,y(SP),#0,#31,z(SP)       ; y <- y + 10^(-k)y
                            44 AE               0723
                         1E   13                 0725   575         BEQL    20$
   34 AE   1F   44 AE   1F   20                 0727   576         ADDP4   #31,z(SP),#31,y(SP)
24 AE   1F    FABC CF44   1F   22                 072E   577         SUBP4   #31,Table2[R4],#31,x(SP)       ; x <- x - log(1+10^(-k))
                         D1   11                 0737   578         BRB     11$
                            56   D7             0739   579 12$:    DECL    R6                              ; k <- k + 1
  FCAD CF   01   24 AE   1F   37                 073B   580         CMPP4   #31,x(SP),#1,P0                 ; if w = 0 then exitloop
                         C5   12                 0743   581         BNEQ    11$
                                                0745   582 20$:
                            F4 8F   F8           0745   583         ASHP    #INT$K_I_FRACT_D-30,-           ; Shift into result
                   34 AE    1F                  0748   584                 #31,y(SP),-
                            01                  074B   585                 #1,-                            ; (rounded ever so slightly)
                            12                  074C   586                 #INT$K_I_FRACT_D,-
                         1A AE                  074D   587                 INT$P_I_FRACT+res_it(SP)
                            06   1C             074F   588         BVC     21$
                      63    01   90             0751   589         MOVB    #1@<4+<1&INT$K_I_FRACT_D>>,(R3) ; Means answer = 1
                        18 AE   B6             0754   590         INCW    INT$W_I_EXP+res_it(SP)          ; Increment exponent (#1)
                                                0757   591
                        18 AE   B6             0757   592 21$:    INCW    INT$W_I_EXP+res_it(SP)          ; Increment exponent (#2)
                                                075A   593         ;
                                                075A   594         ; Put in the correct sign
                                                075A   595         ;
   23 AE    0F   8A                             075A   596         BICB2   #^X0F,INT$K_I_FRACT_D/2+INT$P_I_FRACT+res_it(SP)
23 AE    64 AE   88                             075E   597         BISB2   res_sign(SP),INT$K_I_FRACT_D/2+INT$P_I_FRACT+res_it(SP)
                                                0763   598
                                                0763   599 ;
                                                0763   600 ;       end exponentiate loop
                                                0763   601 ;
   5E    18 AE   9E                             0763   602         MOVAB   res_it(SP),SP                   ; B.O.H.I.
      50    01   D0                             0767   603         MOVL    #1,R0                           ; Success.  Exaltation.
                                                076A   604 ;       BRW     FINISH
                                                076A   605
                                                076A   606 ;
                                                076A   607 ; All done with the hard part.  Now fall through and convert to destination.
                                                076A   608 ;
                                                076A   609
```

```
                     076A   611          .SBTTL  FINISH          Convert to destination type and return
                     076A   612
                     076A   613 ;+
                     076A   614 ;       Enter by branch with (SP) containing the intermediate result
                     076A   615 ;       and 12(AP) pointing to the descriptor for the destination.
                     076A   616 ;       R0 contains routine status
                     076A   617 ;-
                     076A   618
                     076A   619 FINISH:
          02 AE   95 076A   620          TSTB    INT$P_I_FRACT(SP)       ; is fraction zero ?
             04   12 076D   621          BNEQ    8$                      ; no
             6E   B4 076F   622          CLRW    INT$W_I_EXP(SP)         ; force exponent to zero
             0E   11 0771   623          BRB     9$                      ; bypass overflow and underflow
                     0773   624                                          ; checks
                     0773   625 ;+
                     0773   626 ; Check for out-of-range conditions first
                     0773   627 ; We do the check here for all destination type so that we can report
                     0773   628 ; overflow and underflow distinctly.  If we allow the flow to go
                     0773   629 ; directly to various COB$CVTI_x routines, what will be reported
                     0773   630 ; is COB$_INTRESOPE (which is not correct -- we just created the
                     0773   631 ; exception and did not access it -- creating an exception should
                     0773   632 ; distinguish between over_ and under_flow)
                     0773   633 ;-
                     0773   634
                     0773   635 8$:
    0063 8F   6E   B1 0773   636          CMPW    INT$W_I_EXP(SP), #INT$K_I_EXP_HI ; Bigger than max ?
             55   14 0778   637          BGTR    3$                      ; Yes, overflow
    FF9D 8F   6E   B1 077A   638          CMPW    INT$W_I_EXP(SP), #INT$K_I_EXP_LO ; Less than min ?
             56   19 077F   639          BLSS    5$                      ; Yes, underflow
                     0781   640 9$:
                50   DD 0781   641          PUSHL   R0                      ; Save success status
                     0783   642                                          ; Result now at 4(SP)
                     0783   643
          50   0C AC   D0 0783   644          MOVL    12(AP),R0
    1F    00   02 A0   8F 0787   645          CASEB   DSC$B_DTYPE(R0),#0,#31
                   0202' 078C   646 10$:       .WORD   BAD_DT-10$      ;  0 Z
                   0202' 078E   647            .WORD   BAD_DT-10$      ;  1 V
                   0202' 0790   648            .WORD   BAD_DT-10$      ;  2 BU
                   0202' 0792   649            .WORD   BAD_DT-10$      ;  3 WU
                   0202' 0794   650            .WORD   BAD_DT-10$      ;  4 LU
                   0202' 0796   651            .WORD   BAD_DT-10$      ;  5 QU
                   0202' 0798   652            .WORD   BAD_DT-10$      ;  6 B
                   0058' 079A   653            .WORD   20$-10$         ;  7 W
                   0079' 079C   654            .WORD   30$-10$         ;  8 L
                   009A' 079E   655            .WORD   40$-10$         ;  9 Q
                   00BB' 07A0   656            .WORD   50$-10$         ; 10 F
                   00CD' 07A2   657            .WORD   60$-10$         ; 11 D
                   0202' 07A4   658            .WORD   BAD_DT-10$      ; 12 FC
                   0202' 07A6   659            .WORD   BAD_DT-10$      ; 13 DC
                   0202' 07A8   660            .WORD   BAD_DT-10$      ; 14 T
                   0202' 07AA   661            .WORD   BAD_DT-10$      ; 15 NU
                   0202' 07AC   662            .WORD   BAD_DT-10$      ; 16 NL
                   0202' 07AE   663            .WORD   BAD_DT-10$      ; 17 NLO
                   0202' 07B0   664            .WORD   BAD_DT-10$      ; 18 NR
                   0202' 07B2   665            .WORD   BAD_DT-10$      ; 19 NRO
                   0202' 07B4   666            .WORD   BAD_DT-10$      ; 20 NZ
                    0DF' 07B6   667            .WORD   70$-10$         ; 21 P
```

```
                     0202' 07B8    668          .WORD   BAD_DT-10$      ; 22 ZI
                     0202' 07BA    669          .WORD   BAD_DT-10$      ; 23 ZEM
                     0202' 07BC    670          .WORD   BAD_DT-10$      ; 24 DSC
                     0202' 07BE    671          .WORD   BAD_DT-10$      ; 25 OU
                     0202' 07C0    672          .WORD   BAD_DT-10$      ; 26 O
                     0202' 07C2    673          .WORD   BAD_DT-10$      ; 27 G
                     0202' 07C4    674          .WORD   BAD_DT-10$      ; 28 H
                     0202' 07C6    675          .WORD   BAD_DT-10$      ; 29 GC
                     0202' 07C8    676          .WORD   BAD_DT-10$      ; 30 HC
                     0103' 07CA    677          .WORD   80$-10$         ; 31 COBOL intermediate data type
                   01BF   31 07CC    678          BRW     BAD_DT
                          07CF    679
                          07CF    680 ;+
                          07CF    681 ; CIT overflowed.
                          07CF    682 ;-
                          07CF    683 3$:
           00000000'8F    DD 07CF    684          PUSHL   #COB$_INTEXPOVE         ; Overflow signal
                     06   11 07D5    685          BRB     6$                      ; go signal
                          07D7    686
                          07D7    687 ;+
                          07D7    688 ; CIT underflow
                          07D7    689 ;-
                          07D7    690 5$:
           00000000'8F    DD 07D7    691          PUSHL   #COB$_INTEXPUND         ; Underflow signal
      00000000'GF   01    FB 07DD    692 6$:      CALLS   #1,G^LIB$STOP           ; Signal and stop.
                          07E4    693
                          07E4    694 ;+
                          07E4    695 ; Destination is W
                          07E4    696 ;-
                     56   D4 07E4    697 20$:     CLRL    R6                      ; Assume class S
                09   03 A0 91 07E6    698          CMPB    DSC$B_CLASS(R0),#DSC$K_CLASS_SD
                        07 12 07EA    699          BNEQ    21$                     ; Branch if not class SD
                56   08 A0 98 07EC    700          CVTBL   DSC$B_SCALE(R0),R6      ; Get scale factor
                   56   56 CE 07F0    701          MNEGL   R6,R6                   ; Negate scale factor
                57   04 AE 9E 07F3    702 21$:     MOVAB   4(SP),R7                ; Get source address
                58   04 A0 D0 07F7    703          MOVL    DSC$A_POINTER(R0),R8    ; Get destination address
           00000000'GF   16 07FB    704          JSB     G^COB$CVTIW_R8          ; Go to conversion routine
                   50   8E D0 0801    705          MOVL    (SP)+,R0                ; Restore status
                        04 0804    706          RET                             ; Return
                          0805    707
                          0805    708 ;+
                          0805    709 ; Destination is L
                          0805    710 ;-
                     56   D4 0805    711 30$:     CLRL    R6                      ; Assume class S
                09   03 A0 91 0807    712          CMPB    DSC$B_CLASS(R0),#DSC$K_CLASS_SD
                        07 12 080B    713          BNEQ    31$                     ; Branch if not class SD
                56   08 A0 98 080D    714          CVTBL   DSC$B_SCALE(R0),R6      ; Get scale factor
                   56   56 CE 0811    715          MNEGL   R6,R6                   ; Negate scale factor
                57   04 AE 9E 0814    716 31$:     MOVAB   4(SP),R7                ; Get source address
                58   04 A0 D0 0818    717          MOVL    DSC$A_POINTER(R0),R8    ; Get destination address
           00000000'GF   16 081C    718          JSB     G^COB$CVTIL_R8          ; Go to conversion routine
                   50   8E D0 0822    719          MOVL    (SP)+,R0                ; Restore status
                        04 0825    720          RET                             ; Return
                          0826    721
                          0826    722 ;+
                          0826    723 ; Destination is Q
                          0826    724 ;-
```

```
                56   D4   0826    725 40$:    CLRL    R6                              ; Assume class S
 09    03       A0   91   0828    726         CMPB    DSC$B_CLASS(R0),#DSC$K_CLASS_SD
                07   12   082C    727         BNEQ    41$                             ; Branch if not class SD
 56    08       A0   98   082E    728         CVTBL   DSC$B_SCALE(R0),R6              ; Get negative of scale factor
    56           56   CE   0832    729         MNEGL   R6,R6                           ;
 57    04       AE   9E   0835    730 41$:    MOVAB   4(SP),R7                        ; Get source address
 58    04       A0   D0   0839    731         MOVL    DSC$A_POINTER(R0),R8            ; Get destination address
00000000'       GF   16   083D    732         JSB     G^COB$CVTIQ_R8                  ; Go to conversion routine
    50           8E   D0   0843    733         MOVL    (SP)+,R0                        ; Restore status
                     04   0846    734         RET                                     ; Return
                          0847    735
                          0847    736 ;+
                          0847    737 ; Destination is F
                          0847    738 ;-
 56    04       AE   9E   0847    739 50$:    MOVAB   4(SP),R6                        ; Get source address
 57    04       A0   D0   084B    740         MOVL    DSC$A_POINTER(R0),R7            ; Get destination address
00000000'       GF   16   084F    741         JSB     G^COB$CVTIF_R7                  ; Go to conversion routine
    50           8E   D0   0855    742         MOVL    (SP)+,R0                        ; Restore status
                     04   0858    743         RET                                     ; Return
                          0859    744
                          0859    745 ;+
                          0859    746 ; Destination is D
                          0859    747 ;-
 56    04       AE   9E   0859    748 60$:    MOVAB   4(SP),R6                        ; Get source address
 57    04       A0   D0   085D    749         MOVL    DSC$A_POINTER(R0),R7            ; Get destination address
00000000'       GF   16   0861    750         JSB     G^COB$CVTID_R7                  ; Go to conversion routine
    50           8E   D0   0867    751         MOVL    (SP)+,R0                        ; Restore status
                     04   086A    752         RET                                     ; Return
                          086B    753
                          086B    754 ;+
                          086B    755 ; Destination is P
                          086B    756 ;-
                56   D4   086B    757 70$:    CLRL    R6                              ; Assume class S
 09    03       A0   91   086D    758         CMPB    DSC$B_CLASS(R0),#DSC$K_CLASS_SD
                07   12   0871    759         BNEQ    71$                             ; Branch if not class SD
 56    08       A0   98   0873    760         CVTBL   DSC$B_SCALE(R0),R6              ; Get negative of scale factor
    56           56   CE   0877    761         MNEGL   R6,R6                           ;
 57    04       AE   9E   087A    762 71$:    MOVAB   4(SP),R7                        ; Get source address
    58           60   3C   087E    763         MOVZWL  DSC$W_LENGTH(R0),R8             ; Get destination length
 59    04       A0   D0   0881    764         MOVL    DSC$A_POINTER(R0),R9            ; Get destination address
00000000'       GF   16   0885    765         JSB     G^COB$CVTIP_R9                  ; Go to conversion routine
    50           8E   D0   088B    766         MOVL    (SP)+,R0                        ; Restore status
                     04   088E    767         RET                                     ; Return
                          088F    768
                          088F    769 ;+
                          088F    770 ; Destination is intermediate
                          088F    771 ;-
 50    04       A0   D0   088F    772 80$:    MOVL    DSC$A_POINTER(R0),R0            ; Get destination address
 80    04       AE   7D   0893    773         MOVQ    4(SP),(R0)+                     ; Move 8 bytes
 60    0C       AE   D0   0897    774         MOVL    12(SP),(R0)                     ; Move 4 more bytes
    50           8E   D0   089B    775         MOVL    (SP)+,R0                        ; Restore status
                     04   089E    776         RET                                     ; Return
                          089F    777 ;
```

```
                        089F    779          .SBTTL  CONVERT          Internal routine to convert to intermediate
                        089F    780
                        089F    781 ;+
                        089F    782 ;       Call by JSB
                        089F    783 ;       R0 points to descriptor (class = S or SD)
                        089F    784 ;       R1 points to output area (12 bytes)
                        089F    785 ;-
                        089F    786
                        089F    787 CONVERT:
      1F    00    02 A0   8F    089F    788          CASEB   DSC$B_DTYPE(R0),#0,#31  ; Go to proper conversion code
                    00EA' 08A4    789 10$:     .WORD   BAD_DT-10$          ;  0 Z
                    00EA' 08A6    790          .WORD   BAD_DT-10$          ;  1 V
                    00EA' 08A8    791          .WORD   BAD_DT-10$          ;  2 BU
                    00EA' 08AA    792          .WORD   BAD_DT-10$          ;  3 WU
                    00EA' 08AC    793          .WORD   BAD_DT-10$          ;  4 LU
                    00EA' 08AE    794          .WORD   BAD_DT-10$          ;  5 QU
                    00EA' 08B0    795          .WORD   BAD_DT-10$          ;  6 B
                    0043' 08B2    796          .WORD   20$-10$             ;  7 W
                    005C' 08B4    797          .WORD   30$-10$             ;  8 L
                    0075' 08B6    798          .WORD   40$-10$             ;  9 Q
                    008E' 08B8    799          .WORD   50$-10$             ; 10 F
                    009B' 08BA    800          .WORD   60$-10$             ; 11 D
                    00EA' 08BC    801          .WORD   BAD_DT-10$          ; 12 FC
                    00EA' 08BE    802          .WORD   BAD_DT-10$          ; 13 DC
                    00EA' 08C0    803          .WORD   BAD_DT-10$          ; 14 T
                    00EA' 08C2    804          .WORD   BAD_DT-10$          ; 15 NU
                    00EA' 08C4    805          .WORD   BAD_DT-10$          ; 16 NL
                    00EA' 08C6    806          .WORD   BAD_DT-10$          ; 17 NLO
                    00EA' 08C8    807          .WORD   BAD_DT-10$          ; 18 NR
                    00EA' 08CA    808          .WORD   BAD_DT-10$          ; 19 NRO
                    00EA' 08CC    809          .WORD   BAD_DT-10$          ; 20 NZ
                    00A8' 08CE    810          .WORD   70$-10$             ; 21 P
                    00EA' 08D0    811          .WORD   BAD_DT-10$          ; 22 ZI
                    00EA' 08D2    812          .WORD   BAD_DT-10$          ; 23 ZEM
                    00EA' 08D4    813          .WORD   BAD_DT-10$          ; 24 DSC
                    00EA' 08D6    814          .WORD   BAD_DT-10$          ; 25 OU
                    00EA' 08D8    815          .WORD   BAD_DT-10$          ; 26 O
                    00EA' 08DA    816          .WORD   BAD_DT-10$          ; 27 G
                    00EA' 08DC    817          .WORD   BAD_DT-10$          ; 28 H
                    00EA' 08DE    818          .WORD   BAD_DT-10$          ; 29 GC
                    00EA' 08E0    819          .WORD   BAD_DT-10$          ; 30 HC
                    00C4' 08E2    820          .WORD   80$-10$             ; 31 COBOL intermediate data type
              00A7    31  08E4    821          BRW     BAD_DT
                        08E7    822
                        08E7    823 ;+
                        08E7    824 ; Source is W
                        08E7    825 ;-
            56    D4    08E7    826 20$:     CLRL    R6                          ; Assume class S
      09    03 A0   91  08E9    827          CMPB    DSC$B_CLASS(R0),#DSC$K_CLASS_SD
                  04    12  08ED    828          BNEQ    21$                         ; Branch if not class SD
      56    08 A0   98  08EF    829          CVTBL   DSC$B_SCALE(R0),R6          ; Get scale factor
      57    04 A0   D0  08F3    830 21$:     MOVL    DSC$A_POINTER(R0),R7        ; Get source address
         58    51    D0  08F7    831          MOVL    R1,R8                       ; Get destination address
00000000'GF    17    08FA    832          JMP     G^COB$CVTWI_R8              ; Go to conversion routine
                        0900    833
                        0900    834 ;+
                        0900    835 ; Source is L
```

```
                          0900    836 ;-
                          0900    837 30$:    CLRL    R6                              ; Assume class S
          56   D4         0900    838         CMPB    DSC$B_CLASS(R0),#DSC$K_CLASS_SD
09   03 A0   91         0902    839         BNEQ    31$                             ; Branch if not class SD
          04   12         0906    840         CVTBL   DSC$B_SCALE(R0),R6              ; Get scale factor
56   08 A0   98         0908    841 31$:    MOVL    DSC$A_POINTER(R0),R7            ; Get source address
     58   51   D0         090C    842         MOVL    R1,R8                           ; Get destination address
00000000'GF   17         0910    843         JMP     G^COB$CVTLI_R8                  ; Go to conversion routine
                          0913    844
                          0919    845 ;+
                          0919    846 ; Source is Q
                          0919    847 ;-
          56   D4         0919    848 40$:    CLRL    R6                              ; Assume class S
09   03 A0   91         091B    849         CMPB    DSC$B_CLASS(R0),#DSC$K_CLASS_SD
          04   12         091F    850         BNEQ    41$                             ; Branch if not class SD
56   08 A0   98         0921    851         CVTBL   DSC$B_SCALE(R0),R6              ; Get scale factor
57   04 A0   D0         0925    852 41$:    MOVL    DSC$A_POINTER(R0),R7            ; Get source address
     58   51   D0         0929    853         MOVL    R1,R8                           ; Get destination address
00000000'GF   17         092C    854         JMP     G^COB$CVTQI_R8                  ; Go to conversion routine
                          0932    855
                          0932    856 ;+
                          0932    857 ; Source is F
                          0932    858 ;-
56   04 A0   D0         0932    859 50$:    MOVL    DSC$A_POINTER(R0),R6            ; Get source address
     57   51   D0         0936    860         MOVL    R1,R7                           ; Get destination address
00000000'GF   17         0939    861         JMP     G^COB$CVTFI_R7                  ; Go to conversion routine
                          093F    862
                          093F    863 ;+
                          093F    864 ; Source is D
                          093F    865 ;-
 56   04 A0   D0         093F    866 60$:    MOVL    DSC$A_POINTER(R0),R6            ; Get source address
     57   51   D0         0943    867         MOVL    R1,R7                           ; Get destination address
00000000'GF   17         0946    868         JMP     G^COB$CVTDI_R7                  ; Go to conversion routine
                          094C    869
                          094C    870 ;+
                          094C    871 ; Source is P
                          094C    872 ;-
          56   D4         094C    873 70$:    CLRL    R6                              ; Assume class S
09   03 A0   91         094E    874         CMPB    DSC$B_CLASS(R0),#DSC$K_CLASS_SD
          04   12         0952    875         BNEQ    71$                             ; Branch if not class SD
56   08 A0   98         0954    876         CVTBL   DSC$B_SCALE(R0),R6              ; Get scale factor
     57   60   3C         0958    877 71$:    MOVZWL  DSC$W_LENGTH(R0),R7             ; Get source length
58   04 A0   D0         095B    878         MOVL    DSC$A_POINTER(R0),R8            ; Get source address
     59   51   D0         095F    879         MOVL    R1,R9                           ; Get destination address
00000000'GF   17         0962    880         JMP     G^COB$CVTPI_R9                  ; Go to conversion routine
                          0968    881
                          0968    882 ;+
                          0968    883 ; Source is intermediate
                          0968    884 ;-
 50   04 A0   D0         0968    885 80$:    MOVL    DSC$A_POINTER(R0),R0            ; Get source address
0063 8F   60   B1         096C    886         CMPW    INT$W_I_EXP(R0), #INT$K_I_EXP_HI ; Bigger than max ?
          0E   14         0971    887         BGTR    81$                             ; Yes, overflow
FF9D 8F   60   B1         0973    888         CMPW    INT$W_I_EXP(R0), #INT$K_I_EXP_LO ; Less than min ?
          07   19         0978    889         BLSS    81$                             ; Yes, underflow
     81   80   7D         097A    890         MOVQ    (R0)+,(R1)+                     ; Copy 8 bytes
     61   60   D0         097D    891         MOVL    (R0),(R1)                       ; Copy 4 more bytes
                          0980    892         RSB                                     ; Done
```

```
                00000000'8F   DD  0981    893 81$:    PUSHL   #COB$_INTRESOPE          ; Intermediate reserved operand
            00000000'GF   01  FB  0987    894         CALLS   #1,G^LIB$STOP            ; Signal the error
                              098E    895
                              098E    896 ;+
                              098E    897 ; Here if not a supported data type.
                              098E    898 ;-
                00000000'8F   DD  098E    899 BAD_DT: PUSHL   #COB$_INVARG             ; "Invalid argument list"
            00000000'GF   01  FB  0994    900         CALLS   #1,G^LIB$STOP
                              099B    901
                              099B    902         .END
```

```
A$                              = 00000000
B$                              = 00000012
BAD_DT                            0000098E R      02
BASE_IT                         = C0000000
C$                              = 00000024
COB$CVTDI_R7                      ********    X   00
COB$CVTFI_R7                      ********    X   00
COB$CVTID_R7                      ********    X   00
COB$CVTIF_R7                      ********    X   00
COB$CVTIL_R8                      ********    X   00
COB$CVTIP_R9                      ********    X   00
COB$CVTIQ_R8                      ********    X   00
COB$CVTIW_R8                      ********    X   00
COB$CVTLI_R8                      ********    X   00
COB$CVTPI_R9                      ********    X   00
COB$CVTQI_R8                      ********    X   00
COB$CVTWI_R8                      ********    X   00
COB$EXPI                          00000550 RG     02
COB$EXPI_OSE                      00000549 RG     02
COB$_INTEXPOVE                    ********    X   00
COB$_INTEXPUND                    ********    X   00
COB$_INTRESOPE                    ********    X   00
COB$_INVARG                       ********    X   00
COB$_UNDEF_EXP                    ********    X   00
CONVERT                           0000089F R      02
D$                              = 00000030
DSC$A_POINTER                   = 00000004
DSC$B_CLASS                     = 00000003
DSC$B_DTYPE                     = 00000002
DSC$B_SCALE                     = 00000008
DSC$K_CLASS_SD                  = 00000009
DSC$W_LENGTH                    = 00000000
E$                              = 0000003C
ERR_0                             00000525 R      02
ERR_BAD                           00000535 R      02
EXP_IT                          = 0000000C
EXP_J                             00000554 R      02
E_D                             = 00000002
FINISH                            0000076A R      02
INT$K_I_EXP_HI                  = 00000063
INT$K_I_EXP_LO                  = FFFFFF9D
INT$K_I_FRACT_D                 = 00000012
INT$K_I_LEN                     = 0000000C
INT$P_I_FRACT                   = 00000002
INT$W_I_EXP                     = 00000000
LIB$STOP                          ********  X   00
MULBIG                            00000402 R      02
NINES                             000003F2 R      02
OFFSET                          = 00000066
OSE                             = 00000065
P0                                000003F0 R      02
P1                                000003F1 R      02
RES_IT                          = 00000018
RES_SIGN                        = 00000064
TABLE1                          = FFFFFFF0 R      02
TABLE2                          = 000001F0 R      02
X                               = 00000024
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
_COB$CODE	0000099B (2459.)	02 (2.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	33	00:00:00.04	00:00:01.74
Command processing	123	00:00:00.41	00:00:03.53
Pass 1	190	00:00:02.87	00:00:13.45
Symbol table sort	0	00:00:00.17	00:00:00.85
Pass 2	174	00:00:01.43	00:00:04.60
Symbol table output	9	00:00:00.04	00:00:00.06
Psect synopsis output	3	00:00:00.02	00:00:00.05
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	534	00:00:04.98	00:00:24.28

The working set limit was 1350 pages.
27182 bytes (54 pages) of virtual memory were used to buffer the intermediate code.
There were 20 pages of symbol table space allocated to hold 189 non-local and 46 local symbols.
902 source lines were read in Pass 1, producing 22 object records in Pass 2.
10 pages of virtual memory were used to define 9 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[COBRTL.OBJ]COBRTL.MLB;1	1
_255DUA28:[SYSLIB]STARLET.MLB;2	4
TOTALS (all libraries)	5

203 GETS were required to define 5 macros.

There were no errors, warnings or information messages.

```
MACRO/ENABLE=SUPPRESSION/DISABLE=(GLOBAL,TRACEBACK)/LIS=LIS$:COBEXPI/OBJ=OBJ$:COBEXPI MSRC$:COBEXPI/UPDATE=(ENH$:COBEXPI)+LIB$:COBRT
```

COBDIVQ
LIS

COBFINDNA
LIS

COBDBEXCE
LIS

COBEXPI
LIS

COBDEEDIT
LIS

COBDISPLA
LIS

COBESCGEN
LIS

COBERROR
LIS

COBDHANDL
LIS