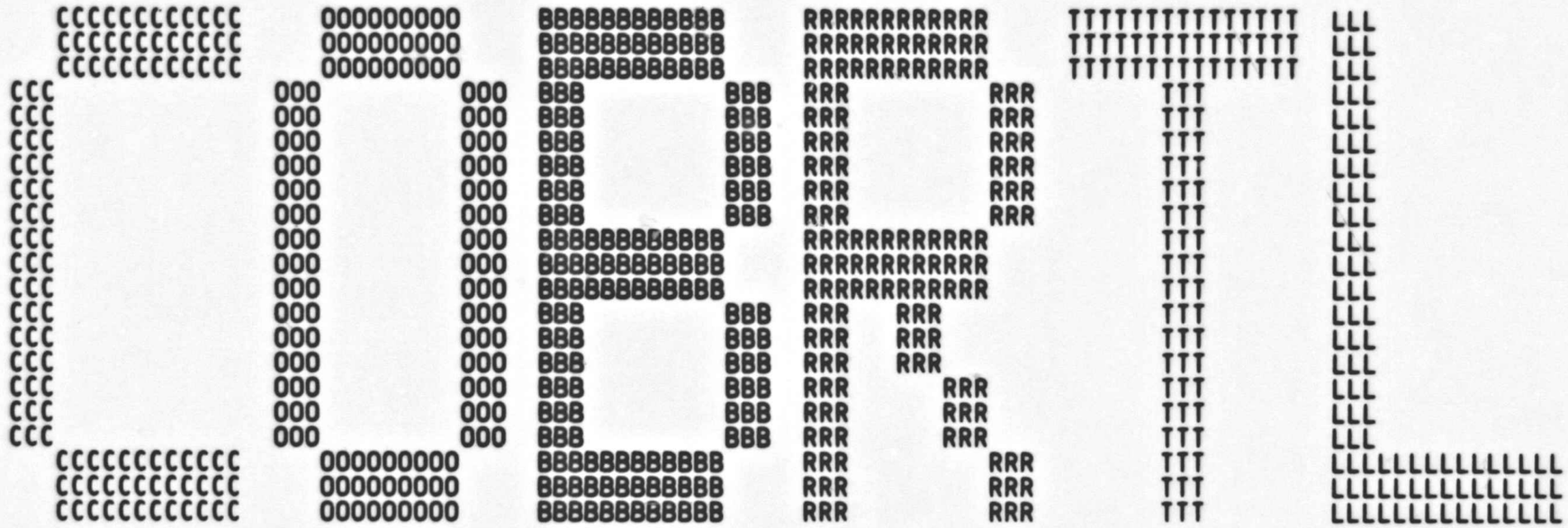
COBRA

```
  CCCCCCCC    OOOOOO    BBBBBBBB     IIIIII    NN      NN  TTTTTTTTTT  EEEEEEEEEE  RRRRRRRR
  CCCCCCCC    OOOOOO    BBBBBBBB     IIIIII    NN      NN  TTTTTTTTTT  EEEEEEEEEE  RRRRRRRR
CC           OO      OO  BB      BB     II     NN      NN      TT      EE          RR      RR
CC           OO      OO  BB      BB     II     NN      NN      TT      EE          RR      RR
CC           OO      OO  BB      BB     II     NNNN    NN      TT      EE          RR      RR
CC           OO      OO  BB      BB     II     NNNN    NN      TT      EE          RR      RR
CC           OO      OO  BBBBBBBB       II     NN  NN  NN      TT      EEEEEEEE    RRRRRRRR
CC           OO      OO  BBBBBBBB       II     NN  NN  NN      TT      EEEEEEEE    RRRRRRRR
CC           OO      OO  BB      BB     II     NN    NNNN      TT      EE          RR  RR
CC           OO      OO  BB      BB     II     NN    NNNN      TT      EE          RR  RR
CC           OO      OO  BB      BB     II     NN      NN      TT      EE          RR    RR      ....
CC           OO      OO  BB      BB     II     NN      NN      TT      EE          RR    RR      ....
  CCCCCCCC    OOOOOO    BBBBBBBB     IIIIII    NN      NN      TT      EEEEEEEEEE  RR      RR    ....
  CCCCCCCC    OOOOOO    BBBBBBBB     IIIIII    NN      NN      TT      EEEEEEEEEE  RR      RR    ....

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$INTER       COBOL Intermediate Conversions
0000     2          .IDENT  /1-019/         ; File: COBINTER.MAR
0000     3
0000     4 ;
0000     5 ;
0000     6 ;*****************************************************************************
0000     7 ;*                                                                           *
0000     8 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     9 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000    10 ;*  ALL RIGHTS RESERVED.                                                     *
0000    11 ;*                                                                           *
0000    12 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    13 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    14 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    15 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    16 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    17 ;*  TRANSFERRED.                                                             *
0000    18 ;*                                                                           *
0000    19 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    20 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    21 ;*  CORPORATION.                                                             *
0000    22 ;*                                                                           *
0000    23 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    24 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    25 ;*                                                                           *
0000    26 ;*                                                                           *
0000    27 ;*****************************************************************************
0000    28 ;
0000    29 ;
0000    30 ;
0000    31 ; FACILITY: COBOL DATA TYPE CONVERSIONS
0000    32 ;++
0000    33 ; ABSTRACT:
0000    34 ;       This module contains all of the data type conversions
0000    35 ;       to and from intermediate temporary for COBOL.
0000    36 ;
0000    37 ;
0000    38 ;--
0000    39 ;
0000    40 ; VERSION: 1
0000    41 ;
0000    42 ; HISTORY:
0000    43 ;
0000    44 ; AUTHOR:
0000    45 ;       John Sauter, 02-JAN-1979
0000    46 ;
0000    47 ; MODIFIED BY:
0000    48 ;
0000    49 ;
0000    50 ;
```

```
0000     52          .SBTTL  HISTORY         ; Detailed Current Edit History
0000     53
0000     54
0000     55 ; Edit History for Version 1 of COBINTER
0000     56 ;
0000     57 ; 1-001 - Original.  JBS 02-JAN-1979
0000     58 ; 1-002 - Continue development.  JBS 12-JAN-1979
0000     59 ; 1-003 - Rearrange intermediate form to put fraction before exponent.
0000     60 ;         JBS 16-JAN-1979
0000     61 ; 1-004 - Remove the arithmetic routines.  DGP 23-Jan-79
0000     62 ; 1-005 - Remove the constants which refer to the Intermediate data type and
0000     63 ;         put them in a Require file - COBPARINT.MDL.  DGP 29-Jan-79
0000     64 ; 1-006 - Change the references to the Intermediate data type parameters from
0000     65 ;         COB$ to INT$.  DGP 30-Jan-79
0000     66 ; 1-007 - Declare externals explicitly.  JBS 06-FEB-1979
0000     67 ; 1-008 - Bug fixing and cleanup.  MLJ 10-Mar-1979
0000     68 ; 1-009 - 19 digit temps and bug fixing.  MLJ 13-Mar-1979
0000     69 ; 1-010 - Correct problem with high order longword in CVTIQ and CVTRIQ.  MLJ 22-Mar-
0000     70 ; 1-011 - Comment out ASHP that specified rounding in CVTIP.  WPS 11-June-1979.
0000     71 ; 1-012 - Correct DIVP rounding toward 0 in conversions to quad.  PDG 12-Jul-1979
0000     72 ; 1-013 - Get edit history in synch with ident statement.
0000     73 ;         RKR 04-Sept-1979
0000     74 ; 1-014 - Set INT$P_I_FRAC to 2.  RKR 11-Sept-79
0000     75 ; 1-015 - Change all references to FOR$CNV_IN_DEFG to OTS$_CVT_T_D
0000     76 ;         RKR 27-SEPT-79
0000     77 ; 1-016 - Change all references to FOR$CNV_OUT_E to COB$CNVOUT.
0000     78 ;         RKR 27-SEPT-79
0000     79 ; 1-017 - Make INT$$NORMALIZE local.  Cosmetic changes.  RKR 21-OCT-79
0000     80 ; 1-018 - Change CIT reserved operand detection logic.  RKR 30-OCT-79
0000     81 ; 1-019 - Fix normalization problem in CVTPI.  MLJ 21-Nov-1979
0000     82 ;
```

```
                        0000     84          .SBTTL  DECLARATIONS
                        0000     85
                        0000     86          .DSABL  GBL
                        0000     87
                        0000     88 ;
                        0000     89 ; INCLUDE FILES:
                        0000     90 ;
                        0000     91          $INTDEF                         ; Intermediate data type format
                        0000     92          $DSCDEF                         ; Descriptor definitions
                        0000     93 ;
                        0000     94 ; EXTERNAL SYMBOLS:
                        0000     95 ;
                        0000     96          .EXTRN  LIB$SIGNAL      ; Signal an error
                        0000     97          .EXTRN  LIB$STOP        ; Signal a fatal error
                        0000     98 ;+
                        0000     99 ; The following are error codes for LIB$SIGNAL and LIB$STOP
                        0000    100 ;-
                        0000    101          .EXTRN  COB$_INTEXPOVE  ; Intermediate exponent overflow
                        0000    102          .EXTRN  COB$_INTEXPUND  ; Intermediate exponent underflow
                        0000    103          .EXTRN  COB$_INTRESOPE  ; Intermediate reserved operand
                        0000    104          .EXTRN  COB$_INVDECDIG  ; Invalid decimal digit
                        0000    105          .EXTRN  OTS$_FATINTERR  ; Fatal internal OTS error
                        0000    106 ;+
                        0000    107 ; The following subroutines are used for conversion between floating
                        0000    108 ; point and text.
                        0000    109 ;-
                        0000    110          .EXTRN  COB$CNVOUT      ; Convert floating to E format
                        0000    111          .EXTRN  OTS$CVT_T_D     ; Convert text to floating
                        0000    112 ;
                        0000    113 ; MACROS:
                        0000    114 ;
                        0000    115 ;       NONE
                        0000    116 ;
                        0000    117 ; PSECT DECLARATIONS:
                    00000000    118          .PSECT  _COB$CODE       PIC, SHR, LONG, EXE, NOWRT
                        0000    119
                        0000    120 ;
                        0000    121 ; EQUATED SYMBOLS:
                    00000002    122          INT$P_I_FRACT = 2                       ; Temporary until Packed supported in MDL
                        0000    123                                                  ; Fraction field offset
```

```
                     0000   125 ;
                     0000   126 ; OWN STORAGE:
                     0000   127 ;
                     0000   128 ;+
                     0000   129 ; The following constant has the value 2**32.  It is used for scaling
                     0000   130 ; the high 32 bits and for compensating for unsigned arithmetic.
                     0000   131 ;-
6C 29 67 49 29 04    0000   132 BIAS:    .PACKED 4294967296              ; 2**32
                     0006   133 ;+
                     0006   134 ; The following constant is 2**32-1.  It's subtracted from negative numbers
                     0006   135 ; when converting to quad, to compensate for DIVP truncating toward zero.
                     0006   136 ;-
5C 29 67 49 29 04    0006   137 BIAS_1: .PACKED 4294967295               ; 2**32-1
           0000000A  000C   138 BIAS_DIGITS=10                          ; Number of digits in 2**32
                     000C   139 ;
                     000C   140 ; Note that the Quadword-to-Intermediate conversion routine knows that
                     000C   141 ; the intermediate-form fraction has 18 digits.  This is because a
                     000C   142 ; quadword has 18 or (rarely) 19 digits, and the code is significantly
                     000C   143 ; more efficient if it can shift by 0 or (rarely) 1 digit rather than
                     000C   144 ; computing how much to shift, as the Packed-to-Intermediate conversion
                     000C   145 ; routine must.
                     000C   146 ;
                     000C   147 ; Similarly, NORMALIZE and CVTTI manipulate the intermediate form
                     000C   148 ; internals directly.
                     000C   149 ;--
```

```
                                000C   151          .SBTTL  NORMALIZE
                                000C   152 ;
                                000C   153 ; This is the code to normalize an intermediate-form number.  It is
                                000C   154 ; branched to from many places in the code.  It expects the
                                000C   155 ; unnormalized intermediate-form number on the stack.  The normalized
                                000C   156 ; number is stored in (R8).
                                000C   157 ;
                                000C   158 INT$$NORMALIZE:
                          02 AE 95     000C   159          TSTB    INT$P_I_FRACT(SP)       ; Is the number already normal?
                             0A 13     000F   160          BEQL    2$                      ; Maybe not.  (May be zero.)
                                0011   161 ;
                                0011   162 ; Come here if the number is already normal.  Copy it to the result
                                0011   163 ; area and then check for underflow and overflow.  Note that this code
                                0011   164 ; knows the intermediate-form number is 12 bytes long.
                                0011   165 ;
                       68       6E 7D 0011   166          MOVQ    (SP),(R8)               ; Copy the number
                    08 A8    08 AE D0 0014   167          MOVL    8(SP),8(R8)             ;
                             23 11   0019   168          BRB     4$                      ; Check for fault and return
                                001B   169 ;
                                001B   170 ; The number is unnormalized or it is zero.  If all of the digits of
                                001B   171 ; the fraction are zero then the number is zero.
                                001B   172 ;
                 02 AE       0A 00 3B 001B   173 2$:      SKPC    #0,#INT$K_I_FRACT_L,INT$P_I_FRACT(SP)
                                0020   174                                              ; Find first non-zero byte
                    54       0A 50 C3 0020   175          SUBL3   R0,#INT$K_I_FRACT_L,R4  ; Convert to byte offset from base
                             54 54 C0 0024   176          ADDL2   R4,R4                   ; Convert bytes to digits
                       61    F0 8F 93 0027   177          BITB    #^XF0,(R1)              ; Is high digit non-zero?
                             02 13   002B   178          BEQL    3$                      ; Br if so
                             54 D7   002D   179          DECL    R4                      ; No, shift one less
    12    00    02 AE       12 54 F8 002F   180 3$:      ASHP    R4,#INT$K_I_FRACT_D,INT$P_I_FRACT(SP),#0, -
                          02 A8        0036
                                0038   181                      #INT$K_I_FRACT_D,INT$P_I_FRACT(R8)
                                0038   182                                              ; Normalize and store fraction
                             19 13   0038   183          BEQL    6$                      ; Br if fraction is zero
                       68    6E 54 A3 003A   184          SUBW3   R4,INT$W_I_EXP(SP),INT$W_I_EXP(R8)
                                003E   185                                              ; Adjust and store exponent
                                003E   186 ;
                                003E   187 ; Now check for overflow and underflow.
                                003E   188 ;
                    0063 8F     68 B1 003E   189 4$:      CMPW    INT$W_I_EXP(R8),#INT$K_I_EXP_HI
                                0043   190                                              ; Is exponent too big?
                             17 14   0043   191          BGTR    7$                      ; Yes, give message
                    FF9D 8F     68 B1 0045   192          CMPW    INT$W_I_EXP(R8),#INT$K_I_EXP_LO
                                004A   193                                              ; Is exponent too small?
                             18 19   004A   194          BLSS    8$                      ; Yes, give message
                          50 01 D0   004C   195          MOVL    #1,R0                   ; No, indicate success
                          5E 0C C0   004F   196 5$:      ADDL2   #INT$K_I_LEN,SP         ; Remove temp from stack
                             05      0052   197          RSB                             ; Return to caller.
                                0053   198 ;
                                0053   199 ; Come here if the fraction is zero to store a clean exponent.
                                0053   200 ;
                             68 B4   0053   201 6$:      CLRW    INT$W_I_EXP(R8)         ; Set exponent to zero
                          50 01 D0   0055   202          MOVL    #1,R0                   ; Indicate success
                          5E 0C C0   0058   203          ADDL2   #INT$K_I_LEN,SP         ; Remove temp from stack
                             05      005B   204          RSB                             ; Return to caller.
                                005C   205 ;
                                005C   206 ; Come to 7$ on overflow.  Signal INTEXPOVE.
```

```
                        005C   207 ; Come to 8$ on underflow.  Signal INTEXPUND.
                        005C   208 ;
      00000000'8F   DD  005C   209 7$:     PUSHL   #COB$_INTEXPOVE          ; Intermediate exponent overflow
                    06  11  0062   210         BRB     9$                   ; Print message and return to caller
      00000000'8F   DD  0064   211 8$:     PUSHL   #COB$_INTEXPUND          ; Intermediate exponent underflow
02 A8    12   00    F9  006A   212 9$:     CVTLP   #0,#INT$K_I_FRACT_D,INT$P_I_FRACT(R8)
                        006F   213                                           ; Set fraction to zero
00000000'GF   01    FB  006F   214         CALLS   #1,G^LIB$SIGNAL          ; Print message for user
                    50  D4  0076   215         CLRL    R0                   ; Indicate failure
                    D5  11  0078   216         BRB     5$                   ; Return to caller
```

```
                     007A   218          .SBTTL  COB$CVTWI_R8    Convert Word to Intermediate
                     007A   219
                     007A   220 ;++
                     007A   221 ; FUNCTIONAL DESCRIPTION:
                     007A   222 ;
                     007A   223 ;       Convert a 16-bit (word) integer to intermediate
                     007A   224 ;
                     007A   225 ; CALLING SEQUENCE:
                     007A   226 ;
                     007A   227 ;       JSB COB$CVTWI_R8 (scale.rl.v, src.rw.r, dst.wx.r)
                     007A   228 ;
                     007A   229 ;       Arguments are passed in R6, R7 and R8.
                     007A   230 ;
                     007A   231 ; INPUT PARAMETERS:
                     007A   232 ;
                     007A   233 ;       SCALE.rl.v              The power of ten by which the internal
                     007A   234 ;                               representation of the source must be
                     007A   235 ;                               multiplied to scale the same as the
                     007A   236 ;                               internal representation of the dest.
                     007A   237 ;       SRC.rw.r                The number to be converted
                     007A   238 ;
                     007A   239 ; IMPLICIT INPUTS:
                     007A   240 ;
                     007A   241 ;       All of the trap bits in the PSL are assumed off.
                     007A   242 ;
                     007A   243 ; OUTPUT PARAMETERS:
                     007A   244 ;
                     007A   245 ;       DST.wx.r                The place to store the converted number
                     007A   246 ;
                     007A   247 ; IMPLICIT OUTPUTS:
                     007A   248 ;
                     007A   249 ;       NONE
                     007A   250 ;
                     007A   251 ; FUNCTION VALUE:
                     007A   252 ;
                     007A   253 ;       1 = SUCCESS, 0 = FAILURE
                     007A   254 ;
                     007A   255 ; SIDE EFFECTS:
                     007A   256 ;
                     007A   257 ;       Destroys registers R0 through R8.
                     007A   258 ;
                     007A   259 ;--
                     007A   260
                     007A   261 COB$CVTWI_R8::
          5E   0C    C2   007A   262          SUBL2   #INT$K_I_LEN,SP         ; Make room on stack for result
          50   67    32   007D   263          CVTWL   (R7),R0                 ; Get source as a longword
02 AE     12   50    F9   0080   264          CVTLP   R0,#INT$K_I_FRACT_D,INT$P_I_FRACT(SP)
                          0085   265                                          ; Store as fraction part
   6E     12   56    A1   0085   266          ADDW3   R6,#INT$K_I_FRACT_D,INT$W_I_EXP(SP)
                          0089   267                                          ; Compute and store exponent
               FF80    31   0089   268          BRW     INT$$NORMALIZE          ; Normalize and return.
```

```
                     008C   270          .SBTTL  COB$CVTLI_R8    Convert Longword to Intermediate
                     008C   271
                     008C   272 ;++
                     008C   273 ; FUNCTIONAL DESCRIPTION:
                     008C   274 ;
                     008C   275 ;       Convert a 32-bit (longword) integer to intermediate
                     008C   276 ;
                     008C   277 ; CALLING SEQUENCE:
                     008C   278 ;
                     008C   279 ;       JSB COB$CVTLI_R8 (scale.rl.v, src.rl.r, dst.wx.r)
                     008C   280 ;
                     008C   281 ;       Arguments are passed in R6, R7 and R8.
                     008C   282 ;
                     008C   283 ; INPUT PARAMETERS:
                     008C   284 ;
                     008C   285 ;       SCALE.rl.v              The power of ten by which the internal
                     008C   286 ;                               representation of the source must be
                     008C   287 ;                               multiplied to scale the same as the
                     008C   288 ;                               internal representation of the dest.
                     008C   289 ;       SRC.rl.r                The number to be converted
                     008C   290 ;
                     008C   291 ; IMPLICIT INPUTS:
                     008C   292 ;
                     008C   293 ;       All of the trap bits in the PSL are assumed off.
                     008C   294 ;
                     008C   295 ; OUTPUT PARAMETERS:
                     008C   296 ;
                     008C   297 ;       DST.wx.r                The place to store the converted number
                     008C   298 ;
                     008C   299 ; IMPLICIT OUTPUTS:
                     008C   300 ;
                     008C   301 ;       NONE
                     008C   302 ;
                     008C   303 ; FUNCTION VALUE:
                     008C   304 ;
                     008C   305 ;       1 = SUCCESS, 0 = FAILURE
                     008C   306 ;
                     008C   307 ; SIDE EFFECTS:
                     008C   308 ;
                     008C   309 ;       Destroys registers R0 through R8.
                     008C   310 ;
                     008C   311 ;--
                     008C   312
                     008C   313 COB$CVTLI_R8::
          5E   0C    C2   008C   314          SUBL2   #INT$K_I_LEN,SP          ; Make room on stack for result
02 AE     12   67    F9   008F   315          CVTLP   (R7),#INT$K_I_FRACT_D,INT$P_I_FRACT(SP)
                          0094   316                                           ; Store source as fraction part
   6E     12   56    A1   0094   317          ADDW3   R6,#INT$K_I_FRACT_D,INT$W_I_EXP(SP)
                          0098   318                                           ; Compute and store exponent
          FF71  31    0098   319          BRW     INT$$NORMALIZE           ; Normalize and return
```

```
                     009B    321          .SBTTL  COB$CVTQI_R8    Convert Quadword to Intermediate
                     009B    322
                     009B    323 ;++
                     009B    324 ; FUNCTIONAL DESCRIPTION:
                     009B    325 ;
                     009B    326 ;       Convert a 64-bit (quadword) integer to intermediate
                     009B    327 ;
                     009B    328 ; CALLING SEQUENCE:
                     009B    329 ;
                     009B    330 ;       JSB COB$CVTQI_R8 (scale.rl.v, src.rq.r, dst.wx.r)
                     009B    331 ;
                     009B    332 ;       Arguments are passed in R6, R7 and R8.
                     009B    333 ;
                     009B    334 ; INPUT PARAMETERS:
                     009B    335 ;
                     009B    336 ;       SCALE.rl.v              The power of ten by which the internal
                     009B    337 ;                               representation of the source must be
                     009B    338 ;                               multiplied to scale the same as the
                     009B    339 ;                               internal representation of the dest.
                     009B    340 ;       SRC.rq.r                The number to be converted
                     009B    341 ;
                     009B    342 ; IMPLICIT INPUTS:
                     009B    343 ;
                     009B    344 ;       All of the trap bits in the PSL are assumed off.
                     009B    345 ;
                     009B    346 ; OUTPUT PARAMETERS:
                     009B    347 ;
                     009B    348 ;       DST.wx.r                The place to store the converted number
                     009B    349 ;
                     009B    350 ; IMPLICIT OUTPUTS:
                     009B    351 ;
                     009B    352 ;       NONE
                     009B    353 ;
                     009B    354 ; FUNCTION VALUE:
                     009B    355 ;
                     009B    356 ;       1 = SUCCESS, 0 = FAILURE
                     009B    357 ;
                     009B    358 ; SIDE EFFECTS:
                     009B    359 ;
                     009B    360 ;       Destroys registers R0 through R8.
                     009B    361 ;
                     009B    362 ;--
                     009B    363
                     009B    364 COB$CVTQI_R8::
              5E     0C   C2 009B    365          SUBL2   #INT$K_I_LEN,SP         ; Make room on stack for result
04 A7   67    01     1F   EC 009E    366          CMPV    #31,#1,(R7),4(R7)       ; Is number in longword range?
                     0C   12 00A4    367          BNEQ    1$                      ; Br if not to do slower code
     02 AE    12     67   F9 00A6    368          CVTLP   (R7),#INT$K_I_FRACT_D,INT$P_I_FRACT(SP)
                     00AB    369                                                  ; Store source as fraction part
        6E    12     56   A1 00AB    370          ADDW3   R6,#INT$K_I_FRACT_D,INT$W_I_EXP(SP)
                     00AF    371                                                  ; Compute and store exponent
              FF5A   31   00AF    372          BRW     INT$$NORMALIZE          ; Normalize and return
                     00B2    373 ;
                     00B2    374 ; Come here if the number is not in longword range.
                     00B2    375 ;
              5E     14   C2 00B2    376 1$:      SUBL2   #20,SP                  ; Allocate temp space
     6E   0A   04 A7   F9 00B5    377          CVTLP   4(R7),#10,(SP)          ; Convert high order longword
```

```
   13  6E  0A  FF41 CF   0A   25  00BA   378          MULP    #BIAS_DIGITS,BIAS,#10,(SP),#19,8(SP)
                     08 AE        00C2
                                  00C4   379                                  ; Multiply by 2**32
                 6E  0A  67   F9  00C4   380          CVTLP   (R7),#10,(SP)       ; Convert low order longword
                         07   18  00C8   381          BGEQ    2$                  ; Br if nonnegative
         6E  0A  FF31 CF   0A   20  00CA   382          ADDP4   #BIAS_DIGITS,BIAS,#10,(SP)
                                  00D1   383                                  ; Correct for signed conversion
           08 AE  13  6E   0A   20  00D1   384  2$:     ADDP4   #10,(SP),#19,8(SP)  ; Sum low and high order parts
                                  00D7   385  ;
                                  00D7   386  ; Try to fit the number in 18 digits.  If this succeeds, we can use the
                                  00D7   387  ; normalize exit.  If not, we must truncate one low order digit.
                                  00D7   388  ;
16 AE   12  00  63  13   00   F8  00D7   389          ASHP    #0,#19,(R3),#0,#INT$K_I_FRACT_D,INT$P_I_FRACT+20(SP)
                                  00DF   390                                  ; Try to fit in 18 digits
                         0B   1D  00DF   391          BVS     3$                  ; Fail, do special work.
               14 AE  12  56   A1  00E1   392          ADDW3   R6,#INT$K_I_FRACT_D,INT$W_I_EXP+20(SP)
                                  00E6   393                                  ; Exponent with scale factor
                     5E   14   C0  00E6   394          ADDL2   #20,SP              ; Remove local temps
                        FF20   31  00E9   395          BRW     INT$$NORMALIZE      ; Normalize and return
                                  00EC   396  ;
                                  00EC   397  ; Come here if we have 19 digits.  We must truncate to 18 digits and
                                  00EC   398  ; compensate for the right shift in the exponent.
                                  00EC   399  ;
63   12   00   61   13    FF 8F   F8  00EC   400  3$:     ASHP    #-1,#19,(R1),#0,#INT$K_I_FRACT_D,(R3)
                                  00F4   401                                  ; Drop low digit of result
               14 AE  13   56   A1  00F4   402          ADDW3   R6,#INT$K_I_FRACT_D+1,INT$W_I_EXP+20(SP)
                                  00F9   403                                  ; Exponent with scale factor
                     5E   14   C0  00F9   404          ADDL2   #20,SP              ; Remove local temps
                        FF0D   31  00FC   405          BRW     INT$$NORMALIZE      ; Normalize and return
```

```
                     00FF    407          .SBTTL  COB$CVTFI_R7    Convert Floating to Intermediate
                     00FF    408
                     00FF    409 ;++
                     00FF    410 ; FUNCTIONAL DESCRIPTION:
                     00FF    411 ;
                     00FF    412 ;       Convert single-precision floating to intermediate
                     00FF    413 ;
                     00FF    414 ; CALLING SEQUENCE:
                     00FF    415 ;
                     00FF    416 ;       JSB COB$CVTFI_R7 (src.rf.r, dst.wx.r)
                     00FF    417 ;
                     00FF    418 ;       Arguments are passed in R6 and R7.
                     00FF    419 ;
                     00FF    420 ; INPUT PARAMETERS:
                     00FF    421 ;
                     00FF    422 ;       SRC.rf.r                The number to be converted
                     00FF    423 ;
                     00FF    424 ; IMPLICIT INPUTS:
                     00FF    425 ;
                     00FF    426 ;       All of the trap bits in the PSL are assumed off.
                     00FF    427 ;
                     00FF    428 ; OUTPUT PARAMETERS:
                     00FF    429 ;
                     00FF    430 ;       DST.wx.r                The place to store the converted number
                     00FF    431 ;
                     00FF    432 ; IMPLICIT OUTPUTS:
                     00FF    433 ;
                     00FF    434 ;       NONE
                     00FF    435 ;
                     00FF    436 ; FUNCTION VALUE:
                     00FF    437 ;
                     00FF    438 ;       1 = SUCCESS, 0 = FAILURE
                     00FF    439 ;
                     00FF    440 ; SIDE EFFECTS:
                     00FF    441 ;
                     00FF    442 ;       Destroys registers R0 through R7.
                     00FF    443 ;
                     00FF    444 ;--
                     00FF    445
                     00FF    446 COB$CVTFI_R7::
          5E   21    C2  00FF    447          SUBL2   #INT$K_I_FRACT_D+7+8,SP ; Result string and value
          6E   66    56  0102    448          CVTFD   (R6),(SP)               ; Put value on the stack
                         0105    449 ;
                         0105    450 ; Build a static descriptor pointing to the text area.
                         0105    451 ;
            08 AE    9F  0105    452          PUSHAB  8(SP)                   ; Address = space left
          7E   01    90  0108    453          MOVB    #DSC$K_CLASS_S,-(SP)    ; Class = static
          7E   0E    90  010B    454          MOVB    #DSC$K_DTYPE_T,-(SP)    ; Type = ASCII text
          7E   19    B0  010E    455          MOVW    #INT$K_I_FRACT_D+7,-(SP) ; String length
                         0111    456 ;
                         0111    457 ; Call COB$CNVOUT.
                         0111    458 ;
               12    9D  0111    459          PUSHL   #INT$K_I_FRACT_D        ; Number of digits in fraction
            04 AE    9F  0113    460          PUSHAB  4(SP)                   ; Address of descriptor
            10 AE    9F  0116    461          PUSHAB  16(SP)                  ; Address of value
00000000'GF    03    FB  0119    462          CALLS   #3,G^COB$CNVOUT ; Convert to E format
            1F   50    E9  0120    463          BLBC    R0,1$                   ; Should never fail
```

```
                          0123    464 ;+
                          0123    465 ; Sitting on the stack now is the result string, at 16(SP).
                          0123    466 ; The string is INT$K_I_FRACT_D+7 characters in length and is in
                          0123    467 ; the following format:
                          0123    468 ;
                          0123    469 ;       +0.ddddddddddddddddddddE+ee
                          0123    470 ;
                          0123    471 ; We will put the digits into the intermediate form number, with the
                          0123    472 ; sign, and the number following the E will go into the exponent.
                          0123    473 ; Note that the number in this form is already normalized.
                          0123    474 ;-
          12 AE   10 AE   90  0123    475         MOVB    16(SP),18(SP)           ; Put sign on top of "."
02 A7   12   12 AE   12   09  0128    476         CVTSP   #INT$K_I_FRACT_D,18(SP),#INT$K_I_FRACT_D,INT$P_I_FRACT(R7)
                          012F    477                                             ; Pack the fraction
   6E   02   26 AE   02   09  012F    478         CVTSP   #2,INT$K_I_FRACT_D+20(SP),#2,(SP)
                          0135    479                                             ; Make packed exponent
        51   6E   02   36  0135    480         CVTPL   #2,(SP),R1                  ; Make exponent binary
                          0139    481                                             ;     (also clears R0)
             67   51   F7  0139    482         CVTLW   R1,INT$W_I_EXP(R7)          ; Store as exponent of int. number
                  50   D6  013C    483         INCL    R0                          ; Indicate success, R0 = 1
             5E   29   C0  013E    484         ADDL2   #INT$K_I_FRACT_D+7+8+8,SP ; Delete stack temps
                  05   0141    485         RSB                                 ; Return to caller
                          0142    486 ;+
                          0142    487 ; Come here on failure of COB$CNVOUT.  Since any single-precision
                          0142    488 ; floating number can be represented in the intermediate form, this
                          0142    489 ; represents an OTS failure.
                          0142    490 ;-
      00000000'8F   DD  0142    491 I$:     PUSHL   #OTS$_FATINTERR             ; Fatal error in OTS
00000000'GF   01   FB  0148    492         CALLS   #1,G^LIB$STOP
```

```
                     014F   494          .SBTTL  COB$CVTDI_R7    Convert Double to Intermediate
                     014F   495
                     014F   496 ;++
                     014F   497 ; FUNCTIONAL DESCRIPTION:
                     014F   498 ;
                     014F   499 ;       Convert double-precision floating to intermediate
                     014F   500 ;
                     014F   501 ; CALLING SEQUENCE:
                     014F   502 ;
                     014F   503 ;       JSB COB$CVTDI_R7 (src.rd.r, dst.wx.r)
                     014F   504 ;
                     014F   505 ;       Arguments are passed in R6 and R7.
                     014F   506 ;
                     014F   507 ; INPUT PARAMETERS:
                     014F   508 ;
                     014F   509 ;       SRC.rd.r                The number to be converted
                     014F   510 ;
                     014F   511 ; IMPLICIT INPUTS:
                     014F   512 ;
                     014F   513 ;       All of the trap bits in the PSL are assumed off.
                     014F   514 ;
                     014F   515 ; OUTPUT PARAMETERS:
                     014F   516 ;
                     014F   517 ;       DST.wx.r                The place to store the converted number
                     014F   518 ;
                     014F   519 ; IMPLICIT OUTPUTS:
                     014F   520 ;
                     014F   521 ;       NONE
                     014F   522 ;
                     014F   523 ; FUNCTION VALUE:
                     014F   524 ;
                     014F   525 ;       1 = SUCCESS, 0 = FAILURE
                     014F   526 ;
                     014F   527 ; SIDE EFFECTS:
                     014F   528 ;
                     014F   529 ;       Destroys registers R0 through R7.
                     014F   530 ;
                     014F   531 ;--
                     014F   532
                     014F   533 COB$CVTDI_R7::
          5E   21    C2  014F   534          SUBL2   #INT$K_I_FRACT_D+7+8,SP ; Result string and value
          6E   66    70  0152   535          MOVD    (R6),(SP)               ; Put value on the stack
                     0155   536 ;
                     0155   537 ; Build a static descriptor pointing to the text area.
                     0155   538 ;
            08 AE    9F  0155   539          PUSHAB  8(SP)                   ; Address = space left
          7E   01    90  0158   540          MOVB    #DSC$K_CLASS_S,-(SP)    ; Class = static
          7E   0E    90  015B   541          MOVB    #DSC$K_DTYPE_T,-(SP)    ; Type = ASCII text
          7E   19    B0  015E   542          MOVW    #INT$K_I_FRACT_D+7,-(SP) ; String length
                     0161   543 ;
                     0161   544 ; Call COB$CNVOUT.
                     0161   545 ;
               12    DD  0161   546          PUSHL   #INT$K_I_FRACT_D        ; Number of digits in fraction
            04 AE    9F  0163   547          PUSHAB  4(SP)                   ; Address of descriptor
            10 AE    9F  0166   548          PUSHAB  16(SP)                  ; Address of value
00000000'GF    03    FB  0169   549          CALLS   #3,G^COB$CNVOUT ; Convert to E format
            1F 50    E9  0170   550          BLBC    R0,1$                   ; Should never fail
```

```
                          0173    551 ;+
                          0173    552 ; Sitting on the stack now is the result string, at 16(SP).
                          0173    553 ; The string is INT$K_I_FRACT_D+7 characters in length and is in
                          0173    554 ; the following format:
                          0173    555 ;
                          0173    556 ;       +0.ddddddddddddddddddddE+ee
                          0173    557 ;
                          0173    558 ; We will put the digits into the intermediate form number, with the
                          0173    559 ; sign, and the number following the E will go into the exponent.
                          0173    560 ; Note that the number in this form is already normalized.
                          0173    561 ;-
          12 AE   10 AE   90  0173    562         MOVB    16(SP),18(SP)                   ; Put sign on top of "."
02 A7   12   12 AE   12   09  0178    563         CVTSP   #INT$K_I_FRACT_D,18(SP),#INT$K_I_FRACT_D,INT$P_I_FRACT(R7)
                          017F    564                                                     ; Pack the digits
   6E   02   26 AE   02   09  017F    565         CVTSP   #2,INT$K_I_FRACT_D+20(SP),#2,(SP)
                          0185    566                                                     ; Make packed exponent
        51   6E   02   36  0185    567         CVTPL   #2,(SP),R1                      ; Make exponent binary
                          0189    568                                                 ;     (also clears R0)
             67   51   F7  0189    569         CVTLW   R1,INT$W_I_EXP(R7)              ; Store as exponent of int. number
                  50   D6  018C    570         INCL    R0                              ; Indicate success, R0 = 1
             5E   29   C0  018E    571         ADDL2   #INT$K_I_FRACT_D+7+8+8,SP       ; Delete stack temps
                  05        0191    572         RSB                                     ; Return to caller
                          0192    573 ;+
                          0192    574 ; Come here on failure of COB$CNVOUT.  Since any double-precision
                          0192    575 ; floating number can be represented in the intermediate form, this
                          0192    576 ; represents an OTS failure.
                          0192    577 ;-
      00000000'8F   DD    0192    578 i$:     PUSHL   #OTS$_FATINTERR                 ; Fatal error in OTS
00000000'GF   01   FB    0198    579         CALLS   #1,G^LIB$STOP
```

```
                        019F    581          .SBTTL  COB$CVTPI_R9    Convert Packed to Intermediate
                        019F    582
                        019F    583 ;++
                        019F    584 ; FUNCTIONAL DESCRIPTION:
                        019F    585 ;
                        019F    586 ;       Convert a packed integer to intermediate
                        019F    587 ;
                        019F    588 ; CALLING SEQUENCE:
                        019F    589 ;
                        019F    590 ;       JSB COB$CVTPI_R9 (scale.rl.v, srclen.rl.v, src.rp.r, dst.wx.r)
                        019F    591 ;
                        019F    592 ;       Arguments are passed in R6, R7, R8 and R9.
                        019F    593 ;
                        019F    594 ; INPUT PARAMETERS:
                        019F    595 ;
                        019F    596 ;       SCALE.rl.v              The power of ten by which the internal
                        019F    597 ;                               representation of the source must be
                        019F    598 ;                               multiplied to scale the same as the
                        019F    599 ;                               internal representation of the dest.
                        019F    600 ;       SRCLEN.rl.v             The number of digits in the source
                        019F    601 ;       SRC.rp.r                The number to be converted
                        019F    602 ;
                        019F    603 ; IMPLICIT INPUTS:
                        019F    604 ;
                        019F    605 ;       All of the trap bits in the PSL are assumed off.
                        019F    606 ;
                        019F    607 ; OUTPUT PARAMETERS:
                        019F    608 ;
                        019F    609 ;       DST.wx.r                The place to store the converted number
                        019F    610 ;
                        019F    611 ; IMPLICIT OUTPUTS:
                        019F    612 ;
                        019F    613 ;       NONE
                        019F    614 ;
                        019F    615 ; FUNCTION VALUE:
                        019F    616 ;
                        019F    617 ;       1 = SUCCESS, 0 = FAILURE
                        019F    618 ;
                        019F    619 ; SIDE EFFECTS:
                        019F    620 ;
                        019F    621 ;       Destroys registers R0 through R9.
                        019F    622 ;
                        019F    623 ;--
                        019F    624
                        019F    625 COB$CVTPI_R9::
                5E  0C  C2      019F    626          SUBL2   #INT$K_I_LEN,SP         ; Make room on stack for result
                12  57  D1      01A2    627          CMPL    R7,#INT$K_I_FRACT_D     ; Will all digits fit?
                    12  14      01A5    628          BGTR    12$                     ; No, must take significant ones.
                                01A7    629 ;+
                                01A7    630 ; Come here if all of the digits of the source will fit in the
                                01A7    631 ; intermediate-form fraction.  This is handled as a special case
                                01A7    632 ; because it occurs often and it saves having to scan the source
                                01A7    633 ; looking for the most significant byte.  (Note that this is done
                                01A7    634 ; in NORMALIZE anyway, so we don't save all that much.)
                                01A7    635 ;-
02 AE   12  00  68  57  00  F8  01A7    636          ASHP    #0,R7,(R8),#0,#INT$K_I_FRACT_D,INT$P_I_FRACT(SP)
                                01AF    637                                          ; Copy source to fraction
```

```
                6E   12   56   A1   01AF   638          ADDW3   R6,#INT$K_I_FRACT_D,INT$W_I_EXP(SP)
                                   01B3   639                                  ; Compute and store exponent
                     58   59   D0   01B3   640          MOVL    R9,R8          ; Point R8 to destination
                          FE53   31   01B6   641          BRW     INT$$NORMALIZE ; Normalize and check for overflow
                                   01B9   642 ;+
                                   01B9   643 ; Come here when the source will not fit in the fraction part of an
                                   01B9   644 ; intermediate form number.  We must scan the source digits looking
                                   01B9   645 ; for the most significant (non-zero) digit.  The result will be
                                   01B9   646 ; normalized, so the NORMALIZE exit will take its fast path, just
                                   01B9   647 ; checking for overflow and underflow.
                                   01B9   648 ;-
             54   57   FF 8F   78   01B9   649 12$:     ASHL    #-1,R7,R4      ; Compute length in bytes =
                          54   D6   01BE   650          INCL    R4             ;     digits/2 + 1
                68   54   00   3B   01C0   651          SKPC    #0,R4,(R8)     ; Find first non-zero byte
                     54   50   C2   01C4   652          SUBL2   R0,R4          ; Convert to byte offset from base
                     54   54   C0   01C7   653          ADDL2   R4,R4          ; Convert bytes to digits
                        02   57   E8   01CA   654          BLBS    R7,125$        ; If number is even-length
                          54   D7   01CD   655          DECL    R4             ;     allow for one high order zero
                 61   F0 8F   93   01CF   656 125$:    BITB    #^XF0,(R1)     ; Is high digit non-zero?
                          02   12   01D3   657          BNEQ    13$            ; Br if not
                          54   D6   01D5   658          INCL    R4             ; Count one more high order zero
                                   01D7   659 ;+
                                   01D7   660 ; R4 now contains the number of high order zeros in the input number.
                                   01D7   661 ; R7-R4 is thus the number of significant digits.  We want to shift the input
                                   01D7   662 ; number into the result such that the most significant digit is in the 18th
                                   01D7   663 ; digit position.  Therefore compute the shift amount as 18-(R7-R4), or
                                   01D7   664 ; R4 - R7 + 18.
                                   01D7   665 ;-
                     54   57   C2   01D7   666 13$:     SUBL2   R7,R4
                     54   12   C0   01DA   667          ADDL2   #18,R4
02 AE    12   00   68   57   54   F8   01DD   668          ASHP    R4,R7,(R8),#0,#INT$K_I_FRACT_D,INT$P_I_FRACT(SP)
                                   01E5   669                                  ; Normalize and store fraction
                     56   54   C2   01E5   670          SUBL2   R4,R6          ; Adjust scale
                          6E   12   56   A1   01E8   671          ADDW3   R6,#INT$K_I_FRACT_D,INT$W_I_EXP(SP)
                                   01EC   672                                  ; Compute and store exponent
                     58   59   D0   01EC   673          MOVL    R9,R8          ; Point R8 to destination
                          FE1A   31   01EF   674          BRW     INT$$NORMALIZE ; Normalize and check for overflow
```

```
                                01F2   676          .SBTTL  COB$CVTIW_R8    Convert Intermediate to Word
                                01F2   677
                                01F2   678 ;++
                                01F2   679 ; FUNCTIONAL DESCRIPTION:
                                01F2   680 ;
                                01F2   681 ;       Convert an intermediate number to word (16-bit) integer.
                                01F2   682 ;
                                01F2   683 ; CALLING SEQUENCE:
                                01F2   684 ;
                                01F2   685 ;       JSB COB$CVTIW_R8 (scale.rl.v, src.rx.r, dst.ww.r)
                                01F2   686 ;
                                01F2   687 ;       Arguments are passed in R6, R7 and R8.
                                01F2   688 ;
                                01F2   689 ; INPUT PARAMETERS:
                                01F2   690 ;
                                01F2   691 ;       SCALE.rl.v              The power of ten by which the internal
                                01F2   692 ;                               representation of the source must be
                                01F2   693 ;                               multiplied to scale the same as the
                                01F2   694 ;                               internal representation of the dest.
                                01F2   695 ;       SRC.rx.r                The number to be converted
                                01F2   696 ;
                                01F2   697 ; IMPLICIT INPUTS:
                                01F2   698 ;
                                01F2   699 ;       All of the trap bits in the PSL are assumed off.
                                01F2   700 ;
                                01F2   701 ; OUTPUT PARAMETERS:
                                01F2   702 ;
                                01F2   703 ;       DST.ww.r                The place to store the converted number
                                01F2   704 ;
                                01F2   705 ; IMPLICIT OUTPUTS:
                                01F2   706 ;
                                01F2   707 ;       NONE
                                01F2   708 ;
                                01F2   709 ; FUNCTION VALUE:
                                01F2   710 ;
                                01F2   711 ;       1 = SUCCESS, 0 = FAILURE
                                01F2   712 ;
                                01F2   713 ; SIDE EFFECTS:
                                01F2   714 ;
                                01F2   715 ;       Destroys registers R0 through R8.
                                01F2   716 ;
                                01F2   717 ;--
                                01F2   718
                                01F2   719 COB$CVTIW_R8::
                          5E    04   C2  01F2   720          SUBL2   #4,SP                       ; Make room for temp storage
                    0063  8F    67   B1  01F5   721          CMPW    INT$W_I_EXP(R7), #INT$K_I_EXP_HI ; Bigger than biggest ?
                          3A    14   1A  01FA   722          BGTR    13$                         ; Yes, overflow
                    FF9D  8F    67   B1  01FC   723          CMPW    INT$W_I_EXP(R7), #INT$K_I_EXP_LO ; Smaller than smallest
                          33    19   02  0201   724          BLSS    13$                         ; Yes, underflow
                       02 A7    95   03  0203   725          TSTB    INT$P_I_FRACT(R7)       ; Is source zero?
                          3D    13   06  0206   726          BEQL    14$                         ; Yes, return zero.
                       56 67    A0   08  0208   727          ADDW2   INT$W_I_EXP(R7),R6      ; No, compute adjusted scale
                       56 12    A2   0B  020B   728          SUBW2   #INT$K_I_FRACT_D,R6
                       50 56    33   0E  020E   729          CVTWB   R6,R0                       ; Shift count a signed byte?
                          19    1D   11  0211   730          BVS     10$                         ; No, special handling
6E    05    00    02 A7    12    56   F8  0213   731          ASHP    R6,#INT$K_I_FRACT_D,INT$P_I_FRACT(R7),#0,#5,(SP)
                                021B   732                                                      ; Scale and integerize fraction
```

```
                 13   1D  021B   733          BVS     11$                     ; Won't fit in 5 digits, error
           51    6E   05   36  021D   734          CVTPL   #5,(SP),R1            ; Make a longword
                                0221   735                                      ;     (also clears R0)
                 68   51   F7  0221   736          CVTLW   R1,(R8)              ; Store result as a word
                      0A   1D  0224   737          BVS     11$                  ; Cannot, indicate an error
                      50   D6  0226   738          INCL    R0                   ; Indicate success, R0 = 1
                      5E   04  C0  0228   739          ADDL2   #4,SP                ; Remove temp from stack
                           05  022B   740          RSB                          ; Return to caller.
                               022C   741 ;+
                               022C   742 ; Come here if the shift count is not in signed byte range.  If it is
                               022C   743 ; positive, an overflow exists.  If it is negative, return zero.
                               022C   744 ;-
                        15 56   0F   E0  022C   745 10$:    BBS     #15,R6,14$           ; Branch if negative
                               0230   746 ;+
                               0230   747 ; Come here if the number cannot be represented in 16 bits.  Give an
                               0230   748 ; error return.
                               0230   749 ;-
                      50   D4  0230   750 11$:    CLRL    R0                   ; Indicate failure
                      5E   04  C0  0232   751          ADDL2   #4,SP                ; Remove temp from stack
                           05  0235   752          RSB                          ; Return to caller.
                               0236   753 ;+
                               0236   754 ; Come here on the reserved operand.  Signal and give an error return.
                               0236   755 ;-
                00000000'8F   DD  0236   756 13$:    PUSHL   #COB$_INTRESOPE      ; Intermediate reserved operand
          00000000'GF   01   FB  023C   757          CALLS   #1,G^LIB$SIGNAL      ; Print message for user
                      EB   11  0243   758          BRB     11$                  ; Indicate failure and return.
                               0245   759 ;+
                               0245   760 ; Come here if the source is zero or if the shift count is too small
                               0245   761 ; for ASHP.  Return zero.
                               0245   762 ;-
                 68   B4  0245   763 14$:    CLRW    (R8)                 ; Store zero in destination
           50    01   D0  0247   764          MOVL    #1,R0                ; Indicate success
           5E    04   C0  024A   765          ADDL2   #4,SP                ; Remove temp from stack
                      05  024D   766          RSB                          ; Return to caller.
```

```
                     024E   768          .SBTTL  COB$CVTRIW_R8   Convert Rounded Intermediate to Word
                     024E   769
                     024E   770 ;++
                     024E   771 ; FUNCTIONAL DESCRIPTION:
                     024E   772 ;
                     024E   773 ;       Convert an intermediate number to word (16-bit) integer.
                     024E   774 ;
                     024E   775 ; CALLING SEQUENCE:
                     024E   776 ;
                     024E   777 ;       JSB COB$CVTRIW_R8 (scale.rl.v, src.rx.r, dst.ww.r)
                     024E   778 ;
                     024E   779 ;       Arguments are passed in R6, R7 and R8.
                     024E   780 ;
                     024E   781 ; INPUT PARAMETERS:
                     024E   782 ;
                     024E   783 ;       SCALE.rl.v              The power of ten by which the internal
                     024E   784 ;                               representation of the source must be
                     024E   785 ;                               multiplied to scale the same as the
                     024E   786 ;                               internal representation of the dest.
                     024E   787 ;       SRC.rx.r                The number to be converted
                     024E   788 ;
                     024E   789 ; IMPLICIT INPUTS:
                     024E   790 ;
                     024E   791 ;       All of the trap bits in the PSL are assumed off.
                     024E   792 ;
                     024E   793 ; OUTPUT PARAMETERS:
                     024E   794 ;
                     024E   795 ;       DST.ww.r                The place to store the converted number
                     024E   796 ;
                     024E   797 ; IMPLICIT OUTPUTS:
                     024E   798 ;
                     024E   799 ;       NONE
                     024E   800 ;
                     024E   801 ; FUNCTION VALUE:
                     024E   802 ;
                     024E   803 ;       1 = SUCCESS, 0 = FAILURE.
                     024E   804 ;
                     024E   805 ; SIDE EFFECTS:
                     024E   806 ;
                     024E   807 ;       Destroys registers R0 through R8.
                     024E   808 ;
                     024E   809 ;--
                     024E   810
                     024E   811 COB$CVTRIW_R8::
                  5E 04 C2  024E   812          SUBL2   #4,SP                   ; Make room for temp storage
            0063  8F 67 B1  0251   813          CMPW    INT$W_I_EXP(R7), #INT$K_I_EXP_HI ; Bigger than biggest ?
                     3A 14  0256   814          BGTR    13$                     ; Yes, overflow
            FF9D  8F 67 B1  0258   815          CMPW    INT$W_I_EXP(R7), #INT$K_I_EXP_LO ; Smaller than smallest
                     33 19  025D   816          BLSS    13$                     ; Yes, underflow
                  02 A7 95  025F   817          TSTB    INT$P_I_FRACT(R7)       ; Is source zero?
                     3D 13  0262   818          BEQL    14$                     ; Yes, return zero.
                  56 67 A0  0264   819          ADDW2   INT$W_I_EXP(R7),R6      ; No, compute adjusted scale
                  56 12 A2  0267   820          SUBW2   #INT$K_I_FRACT_D,R6
                  50 56 33  026A   821          CVTWB   R6,R0                   ; Shift count a signed byte?
                     19 1D  026D   822          BVS     10$                     ; No, special handling
6E 05 05 02 A7    12 56 F8  026F   823          ASHP    R6,#INT$K_I_FRACT_D,INT$P_I_FRACT(R7),#5,#5,(SP)
                             0277   824                                          ; Scale and integerize fraction
```

```
                13   1D  0277   825          BVS     11$                             ; Won't fit in 5 digits, error
          51    6E   05  36  0279   826          CVTPL   #5,(SP),R1                  ; Make a longword
                                0279   827                                          ;     (also clears R0)
                68   51  F7  027D   828          CVTLW   R1,(R8)                     ; Store result as a word
                     0A  1D  0280   829          BVS     11$                         ; Cannot, indicate an error
                     50  D6  0282   830          INCL    R0                          ; Indicate success, R0 = 1
                5E   04  C0  0284   831          ADDL2   #4,SP                       ; Remove temp from stack
                     05  0287   832          RSB                                     ; Return to caller.
                         0288   833 ;+
                         0288   834 ; Come here if the shift count is not in signed byte range.  If it is
                         0288   835 ; positive, an overflow exists.  If it is negative, return zero.
                         0288   836 ;-
             15 56   0F  E0  0288   837 10$:    BBS     #15,R6,14$                  ; Branch if negative
                         028C   838 ;+
                         028C   839 ; Come here if the number cannot be represented in 16 bits.  Give an
                         028C   840 ; error return.
                         028C   841 ;-
                50   D4  028C   842 11$:    CLRL    R0                          ; Indicate failure
                5E   04  C0  028E   843          ADDL2   #4,SP                       ; Remove temp from stack
                     05  0291   844          RSB                                     ; Return to caller.
                         0292   845 ;+
                         0292   846 ; Come here on the reserved operand.  Signal and give an error return.
                         0292   847 ;-
      00000000'8F   DD  0292   848 13$:    PUSHL   #COB$_INTRESOPE             ; Intermediate reserved operand
00000000'GF   01  FB  0298   849          CALLS   #1,G^LIB$SIGNAL             ; Print message for user
                EB   11  029F   850          BRB     11$                         ; Indicate failure and return.
                         02A1   851 ;+
                         02A1   852 ; Come here if the source is zero or if the shift count is too small
                         02A1   853 ; for ASHP.  Return zero.
                         02A1   854 ;-
                68   B4  02A1   855 14$:    CLRW    (R8)                        ; Store zero in destination
                50   01  D0  02A3   856          MOVL    #1,R0                       ; Indicate success
                5E   04  C0  02A6   857          ADDL2   #4,SP                       ; Remove temp from stack
                     05  02A9   858          RSB                                     ; Return to caller.
```

```
                        02AA   860          .SBTTL  COB$CVTIL_R8    Convert Intermediate to Longword
                        02AA   861
                        02AA   862 ;++
                        02AA   863 ; FUNCTIONAL DESCRIPTION:
                        02AA   864 ;
                        02AA   865 ;       Convert an intermediate number to longword (32-bit) integer.
                        02AA   866 ;
                        02AA   867 ; CALLING SEQUENCE:
                        02AA   868 ;
                        02AA   869 ;       JSB COB$CVTIL_R8 (scale.rl.v, src.rx.r, dst.wl.r)
                        02AA   870 ;
                        02AA   871 ;       Arguments are passed in R6, R7 and R8.
                        02AA   872 ;
                        02AA   873 ; INPUT PARAMETERS:
                        02AA   874 ;
                        02AA   875 ;       SCALE.rl.v              The power of ten by which the internal
                        02AA   876 ;                               representation of the source must be
                        02AA   877 ;                               multiplied to scale the same as the
                        02AA   878 ;                               internal representation of the dest.
                        02AA   879 ;       SRC.rx.r                The number to be converted
                        02AA   880 ;
                        02AA   881 ; IMPLICIT INPUTS:
                        02AA   882 ;
                        02AA   883 ;       All of the trap bits in the PSL are assumed off.
                        02AA   884 ;
                        02AA   885 ; OUTPUT PARAMETERS:
                        02AA   886 ;
                        02AA   887 ;       DST.wl.r                The place to store the converted number
                        02AA   888 ;
                        02AA   889 ; IMPLICIT OUTPUTS:
                        02AA   890 ;
                        02AA   891 ;       NONE
                        02AA   892 ;
                        02AA   893 ; FUNCTION VALUE:
                        02AA   894 ;
                        02AA   895 ;       1 = SUCCESS, 0 = FAILURE
                        02AA   896 ;
                        02AA   897 ; SIDE EFFECTS:
                        02AA   898 ;
                        02AA   899 ;       Destroys registers R0 through R8.
                        02AA   900 ;
                        02AA   901 ;--
                        02AA   902
                        02AA   903 COB$CVTIL_R8::
                  5E    08   C2 02AA   904          SUBL2   #8,SP                       ; Make room for temp storage
            0063  8F    67   B1 02AD   905          CMPW    INT$W_I_EXP(R7), #INT$K_I_EXP_HI ; Bigger than biggest ?
                        37   14 02B2   906          BGTR    13$                         ; Yes, overflow
            FF9D  8F    67   B1 02B4   907          CMPW    INT$W_I_EXP(R7), #INT$K_I_EXP_LO ; Smaller than smallest
                        30   19 02B9   908          BLSS    13$                         ; Yes, underflow
                  02 A7 95   95 02BB   909          TSTB    INT$P_I_FRACT(R7)       ; Is source zero?
                        3A   13 02BE   910          BEQL    14$                         ; Yes, return zero.
                  56    67   A0 02C0   911          ADDW2   INT$W_I_EXP(R7),R6      ; No, compute adjusted scale
                  56    12   A2 02C3   912          SUBW2   #INT$K_I_FRACT_D,R6
                  50    56   33 02C6   913          CVTWB   R6,R0                       ; Shift count a signed byte?
                        16   1D 02C9   914          BVS     10$                         ; No, special handling
6E    0A    00    02 A7   12   56   F8 02CB   915          ASHP    R6,#INT$K_I_FRACT_D,INT$P_I_FRACT(R7),#0,#10,(SP)
                              02D3   916                                              ; Scale and integerize fraction
```

```
                10   1D  02D3   917          BVS     11$                             ; Won't fit in 10 digits, error
          68    6E   0A  36  02D5   918          CVTPL   #10,(SP),(R8)               ; Make a longword
                                    02D9   919                                       ;     (also clears R0)
                     0A  1D  02D9   920          BVS     11$                         ; Cannot, indicate an error
                     50  D6  02DB   921          INCL    R0                          ; Indicate success, R0 = 1
                5E   08  C0  02DD   922          ADDL2   #8,SP                       ; Remove temp from stack
                     05  02E0   923          RSB                                     ; Return to caller.
                         02E1   924 ;+
                         02E1   925 ; Come here if the shift count is not in signed byte range.  If it is
                         02E1   926 ; positive, an overflow exists.  If it is negative, return zero.
                         02E1   927 ;-
             15 56   0F  E0  02E1   928 10$:     BBS     #15,R6,14$                  ; Branch if negative
                         02E5   929 ;+
                         02E5   930 ; Come here if the number cannot be represented in 32 bits.  Give an
                         02E5   931 ; error return.
                         02E5   932 ;-
                     50  D4  02E5   933 11$:     CLRL    R0                          ; Indicate failure
                5E   08  C0  02E7   934          ADDL2   #8,SP                       ; Remove temp from stack
                     05  02EA   935          RSB                                     ; Return to caller.
                         02EB   936 ;+
                         02EB   937 ; Come here on the reserved operand.  Signal and give an error return.
                         02EB   938 ;-
          00000000'8F    DD  02EB   939 13$:     PUSHL   #COB$_INTRESOPE             ; Intermediate reserved operand
00000000'GF      01    FB  02F1   940          CALLS   #1,G^LIB$SIGNAL             ; Print message for user
                     EB  11  02F8   941          BRB     11$                         ; Indicate failure and return.
                         02FA   942 ;+
                         02FA   943 ; Come here if the source is zero or if the shift count is too small
                         02FA   944 ; for ASHP.  Return zero.
                         02FA   945 ;-
                     68  D4  02FA   946 14$:     CLRL    (R8)                        ; Store zero in destination
                50   01  D0  02FC   947          MOVL    #1,R0                       ; Indicate success
                5E   08  C0  02FF   948          ADDL2   #8,SP                       ; Remove temp from stack
                     05  0302   949          RSB                                     ; Return to caller.
```

```
                                0303   951          .SBTTL  COB$CVTRIL_R8   Convert Rounded Intermediate to Longword
                                0303   952
                                0303   953 ;++
                                0303   954 ; FUNCTIONAL DESCRIPTION:
                                0303   955 ;
                                0303   956 ;       Convert an intermediate number to longword (32-bit) integer.
                                0303   957 ;
                                0303   958 ; CALLING SEQUENCE:
                                0303   959 ;
                                0303   960 ;       JSB COB$CVTRIL_R8 (scale.rl.v, src.rx.r, dst.wl.r)
                                0303   961 ;
                                0303   962 ;       Arguments are passed in R6, R7 and R8.
                                0303   963 ;
                                0303   964 ; INPUT PARAMETERS:
                                0303   965 ;
                                0303   966 ;       SCALE.rl.v              The power of ten by which the internal
                                0303   967 ;                               representation of the source must be
                                0303   968 ;                               multiplied to scale the same as the
                                0303   969 ;                               internal representation of the dest.
                                0303   970 ;       SRC.rx.r                The number to be converted
                                0303   971 ;
                                0303   972 ; IMPLICIT INPUTS:
                                0303   973 ;
                                0303   974 ;       All of the trap bits in the PSL are assumed off.
                                0303   975 ;
                                0303   976 ; OUTPUT PARAMETERS:
                                0303   977 ;
                                0303   978 ;       DST.wl.r                The place to store the converted number
                                0303   979 ;
                                0303   980 ; IMPLICIT OUTPUTS:
                                0303   981 ;
                                0303   982 ;       NONE
                                0303   983 ;
                                0303   984 ; FUNCTION VALUE:
                                0303   985 ;
                                0303   986 ;       1 = SUCCESS, 0 = FAILURE
                                0303   987 ;
                                0303   988 ; SIDE EFFECTS:
                                0303   989 ;
                                0303   990 ;       Destroys registers R0 through R8.
                                0303   991 ;
                                0303   992 ;--
                                0303   993
                                0303   994 COB$CVTRIL_R8::
                          5E    08   C2  0303   995          SUBL2   #8,SP                           ; Make room for temp storage
                    0063  8F    67   B1  0306   996          CMPW    INT$W_I_EXP(R7), #INT$K_I_EXP_HI ; Bigger than biggest ?
                          37    14       030B   997          BGTR    13$                             ; Yes, overflow
                    FF9D  8F    67   B1  030D   998          CMPW    INT$W_I_EXP(R7), #INT$K_I_EXP_LO ; Smaller than smallest
                          30    19       0312   999          BLSS    13$                             ; Yes, underflow
                       02 A7    95       0314  1000          TSTB    INT$P_I_FRACT(R7)       ; Is source zero?
                          3A    13       0317  1001          BEQL    14$                     ; Yes, return zero.
                    56    67    A0       0319  1002          ADDW2   INT$W_I_EXP(R7),R6      ; No, compute adjusted scale
                    56    12    A2       031C  1003          SUBW2   #INT$K_I_FRACT_D,R6
                    50    56    33       031F  1004          CVTWB   R6,R0                   ; Shift count a signed byte?
                          16    1D       0322  1005          BVS     10$                     ; No, special handling
6E  0A  05  02 A7   12    56    F8       0324  1006          ASHP    R6,#INT$K_I_FRACT_D,INT$P_I_FRACT(R7),#5,#10,(SP)
                                032C  1007                                                  ; Scale and integerize fraction
```

```
                     10   1D  032C  1008          BVS     11$                     ; Won't fit in 10 digits, error
                 68  6E   0A  36  032E  1009          CVTPL   #10,(SP),(R8)        ; Make a longword
                                 0332  1010                                      ;     (also clears R0)
                     0A   1D  0332  1011          BVS     11$                     ; Cannot, indicate an error
                     50   D6  0334  1012          INCL    R0                      ; Indicate success, R0 = 1
                        5E   08  C0  0336  1013          ADDL2   #8,SP           ; Remove temp from stack
                     05   0339  1014          RSB                             ; Return to caller.
                                 033A  1015 ;+
                                 033A  1016 ; Come here if the shift count is not in signed byte range.  If it is
                                 033A  1017 ; positive, an overflow exists.  If it is negative, return zero.
                                 033A  1018 ;-
                              15 56   0F   E0  033A  1019 10$:    BBS     #15,R6,14$              ; Branch if negative
                                 033E  1020 ;+
                                 033E  1021 ; Come here if the number cannot be represented in 32 bits.  Give an
                                 033E  1022 ; error return.
                                 033E  1023 ;-
                     50   D4  033E  1024 11$:    CLRL    R0                      ; Indicate failure
                        5E   08  C0  0340  1025          ADDL2   #8,SP           ; Remove temp from stack
                     05   0343  1026          RSB                             ; Return to caller.
                                 0344  1027 ;+
                                 0344  1028 ; Come here on the reserved operand.  Signal and give an error return.
                                 0344  1029 ;-
                   00000000'8F   DD  0344  1030 13$:    PUSHL   #COB$_INTRESOPE         ; Intermediate reserved operand
                00000000'GF   01   FB  034A  1031          CALLS   #1,G^LIB$SIGNAL         ; Print message for user
                     EB   11   0351  1032          BRB     11$                     ; Indicate failure and return.
                                 0353  1033 ;+
                                 0353  1034 ; Come here if the source is zero or if the shift count is too small
                                 0353  1035 ; for ASHP.  Return zero.
                                 0353  1036 ;-
                     68   D4  0353  1037 14$:    CLRL    (R8)                    ; Store zero in destination
                        50   01   D0  0355  1038          MOVL    #1,R0           ; Indicate success
                        5E   08   C0  0358  1039          ADDL2   #8,SP           ; Remove temp from stack
                     05   035B  1040          RSB                             ; Return to caller.
```

```
                                035C  1042          .SBTTL  COB$CVTIQ_R8    Convert Intermediate to Quadword
                                035C  1043
                                035C  1044 ;++
                                035C  1045 ; FUNCTIONAL DESCRIPTION:
                                035C  1046 ;
                                035C  1047 ;       Convert an intermediate number to quadword (64-bit) integer.
                                035C  1048 ;
                                035C  1049 ; CALLING SEQUENCE:
                                035C  1050 ;
                                035C  1051 ;       JSB COB$CVTIQ_R8 (scale.rl.v, src.rx.r, dst.wq.r)
                                035C  1052 ;
                                035C  1053 ;       Arguments are passed in R6, R7 and R8.
                                035C  1054 ;
                                035C  1055 ; INPUT PARAMETERS:
                                035C  1056 ;
                                035C  1057 ;       SCALE.rl.v              The power of ten by which the internal
                                035C  1058 ;                               representation of the source must be
                                035C  1059 ;                               multiplied to scale the same as the
                                035C  1060 ;                               internal representation of the dest.
                                035C  1061 ;       SRC.rx.r                The number to be converted
                                035C  1062 ;
                                035C  1063 ; IMPLICIT INPUTS:
                                035C  1064 ;
                                035C  1065 ;       All of the trap bits in the PSL are assumed off.
                                035C  1066 ;
                                035C  1067 ; OUTPUT PARAMETERS:
                                035C  1068 ;
                                035C  1069 ;       DST.wq.r                The place to store the converted number
                                035C  1070 ;
                                035C  1071 ; IMPLICIT OUTPUTS:
                                035C  1072 ;
                                035C  1073 ;       NONE
                                035C  1074 ;
                                035C  1075 ; FUNCTION VALUE:
                                035C  1076 ;
                                035C  1077 ;       1 = SUCCESS, 0 = FAILURE
                                035C  1078 ;
                                035C  1079 ; SIDE EFFECTS:
                                035C  1080 ;
                                035C  1081 ;       Destroys registers R0 through R8.
                                035C  1082 ;
                                035C  1083 ;--
                                035C  1084
                                035C  1085 COB$CVTIQ_R8::
                          5E    18    C2  035C  1086          SUBL2   #24,SP                  ; Make room for temp storage
                    0063  8F    67    B1  035F  1087          CMPW    INT$W_I_EXP(R7), #INT$K_I_EXP_HI ; Bigger than biggest ?
                          5E    14    0364  1088          BGTR    13$                     ; Yes, overflow
                    FF9D  8F    67    B1  0366  1089          CMPW    INT$W_I_EXP(R7), #INT$K_I_EXP_LO ; Smaller than smallest
                          57    19    036B  1090          BLSS    13$                     ; Yes, underflow
                       02 A7    95    036D  1091          TSTB    INT$P_I_FRACT(R7)       ; Is source zero?
                          61    13    0370  1092          BEQL    14$                     ; Yes, return zero.
                          56    67    A0  0372  1093          ADDW2   INT$W_I_EXP(R7),R6      ; Compute adjusted scale
                          56    12    A2  0375  1094          SUBW2   #INT$K_I_FRACT_D,R6
                          50    56    33  0378  1095          CVTWB   R6,R0                   ; Shift count in signed byte range?
                                3D    1D  037B  1096          BVS     10$                     ; No, special handling
6E    13    00    02 A7    12    56    F8  037D  1097          ASHP    R6,#INT$K_I_FRACT_D,INT$P_I_FRACT(R7),#0,#19,(SP)
                                      0385  1098                                          ; Scale and integerize fraction
```

```
                        37   1D  0385  1099          BVS     12$                     ; Won't fit in 19 digits, error
                68  6E  13   36  0387  1100          CVTPL   #19,(SP),(R8)          ; Attempt simple conversion
                                0388  1101                                          ;     (also clears R0)
                        0B   1D  0388  1102          BVS     11$                     ; Simple conversion failed.
                                038D  1103 ;+
                                038D  1104 ; Come here if the number can be represented by a longword, which is
                                038D  1105 ; quite common.  We can just spread the sign to the high 32 bits and
                                038D  1106 ; return.
                                038D  1107 ;-
            68  88  E1  8F   78  038D  1108          ASHL    #-31,(R8)+,(R8)         ; Spread sign
                        50   D6  0392  1109          INCL    R0                      ; Indicate success, R0 = 1
                    5E  18   C0  0394  1110          ADDL2   #24,SP                  ; Remove temp storage
                        05   0397  1111          RSB                             ; Return to caller.
                                0398  1112 ;+
                                0398  1113 ; Come here if the number cannot be represented by a longword.  The
                                0398  1114 ; correct low-order 32 bits have already been stored, we must now
                                0398  1115 ; compute the high-order 32 bits.
                                0398  1116 ;-
                    07 09  AE   E9  0398  1117 11$:     BLBC    9(SP),115$              ; Skip if positive
    6E  13  FC65  CF    0A   22  039C  1118          SUBP4   #BIAS_DIGITS,BIAS_1,#19,(SP)  ; Make more negative
13  6E  13  FC58  CF    0A   27  03A3  1119 115$:    DIVP    #BIAS_DIGITS,BIAS,#19,(SP),#19,12(SP)  ; Divide by 2**32
                    0C  AE       03AB
            04  A8  65   13   36  03AD  1120          CVTPL   #19,(R5),4(R8)          ; Convert and store high bits
                                03B2  1121                                          ;     (also clears R0)
                        0A   1D  03B2  1122          BVS     12$                     ; Number too large for a 64-bit integer
                        50   D6  03B4  1123          INCL    R0                      ; Indicate success, R0 = 1
                    5E  18   C0  03B6  1124          ADDL2   #24,SP                  ; Remove temp storage
                        05   03B9  1125          RSB                             ; Return to caller
                                03BA  1126 ;+
                                03BA  1127 ; Come here if the shift count is not in signed byte range.  If it is
                                03BA  1128 ; positive, an overflow exists.  If negative, return zero.
                                03BA  1129 ;-
        15  56  0F   E0  03BA  1130 10$:     BBS     #15,R6,14$              ; Branch if negative
                                03BE  1131 ;+
                                03BE  1132 ; Come here if the number cannot be represented in 64 bits.  Give an
                                03BE  1133 ; error return.
                                03BE  1134 ;-
                    50   D4  03BE  1135 12$:     CLRL    R0                      ; Indicate failure
                5E  18   C0  03C0  1136          ADDL2   #24,SP                  ; Remove temp from stack
                        05   03C3  1137          RSB                             ; Return to caller.
                                03C4  1138 ;+
                                03C4  1139 ; Come here on reserved operand.  Signal an error and give the failure
                                03C4  1140 ; return to the user.
                                03C4  1141 ;-
    00000000'8F   DD  03C4  1142 13$:     PUSHL   #COB$_INTRESOPE         ; Intermediate reserved operand
00000000'GF   01   FB  03CA  1143          CALLS   #1,G^LIB$SIGNAL         ; Print message for user
                    EB   11  03D1  1144          BRB     12$                     ; Give failure return.
                                03D3  1145 ;+
                                03D3  1146 ; Come here if the source is zero or the shift count is too small for
                                03D3  1147 ; ASHP.  Return a zero.
                                03D3  1148 ;-
                68  7C  03D3  1149 14$:     CLRQ    (R8)                    ; Set result to zero
            50  01   D0  03D5  1150          MOVL    #1,R0                   ; Indicate success
                5E  18   C0  03D8  1151          ADDL2   #24,SP                  ; Remove temp from stack
                        05   03DB  1152          RSB                             ; Return to caller.
```

```
                     1154          .SBTTL  COB$CVTRIQ_R8   Convert Rounded Intermediate to Quadword
                     1155
                     1156 ;++
                     1157 ; FUNCTIONAL DESCRIPTION:
                     1158 ;
                     1159 ;       Convert an intermediate number to quadword (64-bit) integer.
                     1160 ;
                     1161 ; CALLING SEQUENCE:
                     1162 ;
                     1163 ;       JSB COB$CVTRIQ_R8 (scale.rl.v, src.rx.r, dst.wq.r)
                     1164 ;
                     1165 ;       Arguments are passed in R6, R7 and R8.
                     1166 ;
                     1167 ; INPUT PARAMETERS:
                     1168 ;
                     1169 ;       SCALE.rl.v              The power of ten by which the internal
                     1170 ;                               representation of the source must be
                     1171 ;                               multiplied to scale the same as the
                     1172 ;                               internal representation of the dest.
                     1173 ;       SRC.rx.r                The number to be converted
                     1174 ;
                     1175 ; IMPLICIT INPUTS:
                     1176 ;
                     1177 ;       All of the trap bits in the PSL are assumed off.
                     1178 ;
                     1179 ; OUTPUT PARAMETERS:
                     1180 ;
                     1181 ;       DST.wq.r                The place to store the converted number
                     1182 ;
                     1183 ; IMPLICIT OUTPUTS:
                     1184 ;
                     1185 ;       NONE
                     1186 ;
                     1187 ; FUNCTION VALUE:
                     1188 ;
                     1189 ;       1 = SUCCESS, 0 = FAILURE
                     1190 ;
                     1191 ; SIDE EFFECTS:
                     1192 ;
                     1193 ;       Destroys registers R0 through R8.
                     1194 ;
                     1195 ;--
                     1196
                     1197 COB$CVTRIQ_R8::
                5E   18   C2  03DC  1198          SUBL2   #24,SP                          ; Make room for temp storage
          0063 8F    67   B1  03DF  1199          CMPW    INT$W_I_EXP(R7), #INT$K_I_EXP_HI ; Bigger than biggest ?
                5E   14   14  03E4  1200          BGTR    13$                             ; Yes, overflow
          FF9D 8F    67   B1  03E6  1201          CMPW    INT$W_I_EXP(R7), #INT$K_I_EXP_LO ; Smaller than smallest
                57   19   19  03EB  1202          BLSS    13$                             ; Yes, underflow
             02 A7   95   95  03ED  1203          TSTB    INT$P_I_FRACT(R7)       ; Is source zero?
                61   13   13  03F0  1204          BEQL    14$                     ; Yes, return zero.
             56 67   A0   A0  03F2  1205          ADDW2   INT$W_I_EXP(R7),R6      ; Compute adjusted scale
             56 12   A2   A2  03F5  1206          SUBW2   #INT$K_I_FRACT_D,R6
             50 56   33   33  03F8  1207          CVTWB   R6,R0                   ; Shift count in signed byte range?
                3D   1D   1D  03FB  1208          BVS     10$                     ; No, special handling
6E   13   05   02 A7   12   56   F8  03FD  1209          ASHP    R6,#INT$K_I_FRACT_D,INT$P_I_FRACT(R7),#5,#19,(SP)
                                  0405  1210                                  ; Scale and integerize fraction
```

```
                     37   1D  0405  1211          BVS     12$                             ; Won't fit in 19 digits, error
                68   6E   13   36  0407  1212          CVTPL   #19,(SP),(R8)              ; Attempt simple conversion
                                 040B  1213                                              ;     (also clears R0)
                     0B   1D  040B  1214          BVS     11$                             ; Simple conversion failed.
                                 040D  1215  ;+
                                 040D  1216  ; Come here if the number can be represented by a longword, which is
                                 040D  1217  ; quite common.  We can just spread the sign to the high 32 bits and
                                 040D  1218  ; return.
                                 040D  1219  ;-
            68   88   E1   8F   78  040D  1220          ASHL    #-31,(R8)+,(R8)         ; Spread sign
                     50   D6  0412  1221          INCL    R0                              ; Indicate success, R0 = 1
                5E   18   C0  0414  1222          ADDL2   #24,SP                          ; Remove temp storage
                     05   0417  1223          RSB                                     ; Return to caller.
                                 0418  1224  ;+
                                 0418  1225  ; Come here if the number cannot be represented by a longword.  The
                                 0418  1226  ; correct low-order 32 bits have already been stored, we must now
                                 0418  1227  ; compute the high-order 32 bits.
                                 0418  1228  ;-
             07 09   AE   E9  0418  1229  11$:     BLBC    9(SP),115$                      ; Skip if positive
      6E 13  FBE5 CF     0A   22  041C  1230          SUBP4   #BIAS_DIGITS,BIAS_1,#19,(SP)  ; Make more negative
13    6E 13  FBD8 CF     0A   27  0423  1231  115$:   DIVP    #BIAS_DIGITS,BIAS,#19,(SP),#19,12(SP)  ; Divide by 2**32
                   0C   AE        042B
         04   A8   65   13   36  042D  1232          CVTPL   #19,(R5),4(R8)           ; Convert and store high bits
                                 0432  1233                                              ;     (also clears R0)
                     0A   1D  0432  1234          BVS     12$                             ; Number too large for a 64-bit integer
                     50   D6  0434  1235          INCL    R0                              ; Indicate success, R0 = 1
                5E   18   C0  0436  1236          ADDL2   #24,SP                          ; Remove temp storage
                     05   0439  1237          RSB                                     ; Return to caller
                                 043A  1238  ;+
                                 043A  1239  ; Come here if the shift count is not in signed byte range.  If it is
                                 043A  1240  ; positive, an overflow exists.  If negative, return zero.
                                 043A  1241  ;-
      15 56    0F   E0  043A  1242  10$:     BBS     #15,R6,14$                      ; Branch if negative
                                 043E  1243  ;+
                                 043E  1244  ; Come here if the number cannot be represented in 64 bits.  Give an
                                 043E  1245  ; error return.
                                 043E  1246  ;-
            50   D4  043E  1247  12$:     CLRL    R0                              ; Indicate failure
         5E   18   C0  0440  1248          ADDL2   #24,SP                          ; Remove temp from stack
                     05   0443  1249          RSB                                     ; Return to caller.
                                 0444  1250  ;+
                                 0444  1251  ; Come here on reserved operand.  Signal an error and give the failure
                                 0444  1252  ; return to the user.
                                 0444  1253  ;-
   00000000'8F   DD  0444  1254  13$:     PUSHL   #COB$_INTRESOPE                 ; Intermediate reserved operand
00000000'GF   01   FB  044A  1255          CALLS   #1,G^LIB$SIGNAL                 ; Print message for user
            EB   11  0451  1256          BRB     12$                             ; Give failure return.
                                 0453  1257  ;+
                                 0453  1258  ; Come here if the source is zero or the shift count is too small for
                                 0453  1259  ; ASHP.  Return a zero.
                                 0453  1260  ;-
            68   7C  0453  1261  14$:     CLRQ    (R8)                            ; Set result to zero
         50   01   D0  0455  1262          MOVL    #1,R0                           ; Indicate success
         5E   18   C0  0458  1263          ADDL2   #24,SP                          ; Remove temp from stack
                     05   045B  1264          RSB                                     ; Return to caller.
```

```
                                045C  1266          .SBTTL  COB$CVTIF_R7    Convert Intermediate to Floating
                                045C  1267
                                045C  1268 ;++
                                045C  1269 ; FUNCTIONAL DESCRIPTION:
                                045C  1270 ;
                                045C  1271 ;       Convert intermediate to single-precision floating
                                045C  1272 ;
                                045C  1273 ; CALLING SEQUENCE:
                                045C  1274 ;
                                045C  1275 ;       JSB COB$CVTIF_R7 (src.rx.r, dst.wf.r)
                                045C  1276 ;
                                045C  1277 ;       Arguments are passed in R6 and R7.
                                045C  1278 ;
                                045C  1279 ; INPUT PARAMETERS:
                                045C  1280 ;
                                045C  1281 ;       SRC.rx.r                The number to be converted
                                045C  1282 ;
                                045C  1283 ; IMPLICIT INPUTS:
                                045C  1284 ;
                                045C  1285 ;       All of the trap bits in the PSL are assumed off.
                                045C  1286 ;
                                045C  1287 ; OUTPUT PARAMETERS:
                                045C  1288 ;
                                045C  1289 ;       DST.wf.r                The place to store the converted number
                                045C  1290 ;
                                045C  1291 ; IMPLICIT OUTPUTS:
                                045C  1292 ;
                                045C  1293 ;       NONE
                                045C  1294 ;
                                045C  1295 ; FUNCTION VALUE:
                                045C  1296 ;
                                045C  1297 ;       1 = SUCCESS, 0 = FAILURE
                                045C  1298 ;
                                045C  1299 ; SIDE EFFECTS:
                                045C  1300 ;
                                045C  1301 ;       Destroys registers R0 through R7.
                                045C  1302 ;
                                045C  1303 ;--
                                045C  1304
                                045C  1305 COB$CVTIF_R7::
          0063 8F    66    B1   045C  1306          CMPW    INT$W_I_EXP(R6), #INT$K_I_EXP_HI ; Bigger than biggest ?
                     6E    14   0461  1307          BGTR    4$                              ; Yes, overflow
          FF9D 8F    66    B1   0463  1308          CMPW    INT$W_I_EXP(R6), #INT$K_I_EXP_LO ; Smaller than smallest
                     67    19   0468  1309          BLSS    4$                              ; Yes, underflow
               5E    21    C2   046A  1310          SUBL2   #INT$K_I_FRACT_D+7+8,SP ; Allocate space for string and result
                                046D  1311 ;+
                                046D  1312 ; Convert the intermediate-form number to text using the following
                                046D  1313 ; format:
                                046D  1314 ;
                                046D  1315 ;       +0.ddddddddddddddddddddE+ee
                                046D  1316 ;
                                046D  1317 ; so that it can be converted by OTS$CVT_T_D.
                                046D  1318 ;-
    0A AE    12    02 A6    12   08   046D  1319          CVTPS   #INT$K_I_FRACT_D,INT$P_I_FRACT(R6), -
                                0474  1320                  #INT$K_I_FRACT_D,10(SP) ; Convert fraction to separate
               08 AE    63    90   0474  1321          MOVB    (R3),8(SP)              ; Move sign over
          09 AE    2E30 8F    B0   0478  1322          MOVW    #^A/0./,9(SP)           ; Start with "+0."
```

```
                50   66   32  047E  1323          CVTWL   INT$W_I_EXP(R6),R0       ; Get exponent
            6E  02   50   F9  0481  1324          CVTLP   R0,#2,(SP)               ; Make a packed number
                     3D   1D  0485  1325          BVS     3$                       ; Should never fail
        1E AE   02   6E   02  08  0487  1326          CVTPS   #2,(SP),#2,INT$K_I_FRACT_D+12(SP)
                                  048D  1327                                       ; Make a separate sign number
              73   45 8F   90  048D  1328          MOVB    #^A/E/,-(R3)             ; Make it "E+ee"
                                  0491  1329 ;+
                                  0491  1330 ; Build the descriptor for OTS$CVT_T_D.
                                  0491  1331 ;-
                08 AE   9F  0491  1332          PUSHAB  8(SP)                    ; Address = area reserved
             7E 01      90  0494  1333          MOVB    #DSC$K_CLASS_S,-(SP)     ; Class = static
             7E 0E      90  0497  1334          MOVB    #DSC$K_DTYPE_T,-(SP)     ; Data type = ASCII text
             7E 19      B0  049A  1335          MOVW    #INT$K_I_FRACT_D+7,-(SP); Length of string
                                  049D  1336 ;+
                                  049D  1337 ; Now call the conversion routine.
                                  049D  1338 ;-
                08 AE   9F  049D  1339          PUSHAB  8(SP)                    ; Address of value
                04 AE   9F  04A0  1340          PUSHAB  4(SP)                    ; Address of descriptor
    00000000'GF    02   FB  04A3  1341          CALLS   #2,G^OTS$CVT_T_D         ; Convert the number
                0D 50   E9  04AA  1342          BLBC    R0,2$                    ; Failure -- must be overflow
              67 08 AE  76  04AD  1343          CVTDF   8(SP),(R7)               ; Store result for user
                     07   1D  04B1  1344          BVS     2$                       ; Conversion failed
                50   01   D0  04B3  1345          MOVL    #1,R0                    ; Indicate success
                5E   29   C0  04B6  1346          ADDL2   #INT$K_I_FRACT_D+7+8+8,SP ; Delete stack temps
                          05  04B9  1347          RSB                              ; Return to caller
                                  04BA  1348 ;+
                                  04BA  1349 ; Come here if the conversion fails.  This means that the intermediate-
                                  04BA  1350 ; form number cannot be represented as a single-precision floating point
                                  04BA  1351 ; number.  Store the reserved operand and return a failure indication.
                                  04BA  1352 ;-
                5E   29   C0  04BA  1353 2$:     ADDL2   #INT$K_I_FRACT_D+7+8+8,SP ; Delete stack temps
              67 01      0F  04BD  1354 1$:     ASHL    #15,#1,(R7)              ; Store floating reserved operand
                     50   D4  04C1  1355          CLRL    R0                       ; Indicate failure
                          05  04C3  1356          RSB                              ; Return to caller
                                  04C4  1357 ;+
                                  04C4  1358 ; Come here if the exponent overflowed 2 decimal digits.  The
                                  04C4  1359 ; intermediate-form number is messed up.  Signal an internal OTS
                                  04C4  1360 ; failure.
                                  04C4  1361 ;-
        00000000'8F    DD   04C4  1362 3$:     PUSHL   #OTS$_FATINTERR          ; OTS internal failure
    00000000'GF    01   FB  04CA  1363          CALLS   #1,G^LIB$STOP
                                  04D1  1364
                                  04D1  1365 ;+
                                  04D1  1366 ; Out-of-range exponenent encountered for a CIT. Complain about it.
                                  04D1  1367 ;-
                                  04D1  1368 4$:
        00000000'8F    DD   04D1  1369          PUSHL   #COB$_INTRESOPE          ; CIT reserved operand signal
    00000000'GF    01   FB  04D7  1370          CALLS   #1, G^LIB$STOP           ; signal and stop.
```

```
                                .SBTTL  COB$CVTID_R7    Convert Intermediate to Double

                        ;++
                        ; FUNCTIONAL DESCRIPTION:
                        ;
                        ;       Convert intermediate to double-precision floating
                        ;
                        ; CALLING SEQUENCE:
                        ;
                        ;       JSB COB$CVTID_R7 (src.rx.r, dst.wd.r)
                        ;
                        ;       Arguments are passed in R6 and R7.
                        ;
                        ; INPUT PARAMETERS:
                        ;
                        ;       SRC.rx.r                The number to be converted
                        ;
                        ; IMPLICIT INPUTS:
                        ;
                        ;       All of the trap bits in the PSL are assumed off.
                        ;
                        ; OUTPUT PARAMETERS:
                        ;
                        ;       DST.wd.r                The place to store the converted number
                        ;
                        ; IMPLICIT OUTPUTS:
                        ;
                        ;       NONE
                        ;
                        ; FUNCTION VALUE:
                        ;
                        ;       1 = SUCCESS, 0 = FAILURE
                        ;
                        ; SIDE EFFECTS:
                        ;
                        ;       Destroys registers R0 through R7.
                        ;
                        ;--

                        COB$CVTID_R7::
     0063 8F    66  B1  04DE  1412          CMPW    INT$W_I_EXP(R6), #INT$K_I_EXP_HI ; Bigger than biggest ?
                6C  14  04E3  1413          BGTR    4$                              ; Yes, overflow
     FF9D 8F    66  B1  04E5  1414          CMPW    INT$W_I_EXP(R6), #INT$K_I_EXP_LO ; Smaller than smallest
                65  19  04EA  1415          BLSS    4$                              ; Yes, underflow
          5E    21  C2  04EC  1416          SUBL2   #INT$K_I_FRACT_D+7+8,SP ; Allocate space for string and result
                        04EF  1417      ;+
                        04EF  1418      ; Convert the intermediate-form number to text using the following
                        04EF  1419      ; format:
                        04EF  1420      ;
                        04EF  1421      ;       +0.ddddddddddddddddddddE+ee
                        04EF  1422      ;
                        04EF  1423      ; so that it can be converted by OTS$CVT_T_D.
                        04EF  1424      ;-
0A AE    12   02 A6    12  08  04EF  1425          CVTPS   #INT$K_I_FRACT_D,INT$P_I_FRACT(R6), -
                        04F6  1426                  #INT$K_I_FRACT_D,10(SP) ; Spread out number
        08 AE   63      90  04F6  1427          MOVB    (R3),8(SP)              ; Move sign over
     09 AE   2E30 8F    B0  04FA  1428          MOVW    #^A/0./,9(SP)           ; Start with "+0."
```

```
                50   66   32  0500  1429          CVTWL   INT$W_I_EXP(R6),R0       ; Get exponent
             6E 02   50   F9  0503  1430          CVTLP   R0,#2,(SP)               ; Make a packed number
                     3B   1D  0507  1431          BVS     3$                       ; Should never fail
          1E AE 02   6E   02  0509  1432          CVTPS   #2,(SP),#2,INT$K_I_FRACT_D+12(SP)
                          050F  1433                                               ; Make a separate sign number
               73 45 8F   90  050F  1434          MOVB    #^A/E/,-(R3)             ; Make it "E+ee"
                          0513  1435 ;+
                          0513  1436 ; Build the descriptor for OTS$CVT_T_D.
                          0513  1437 ;-
                08 AE 9F  0513  1438          PUSHAB  8(SP)                    ; Address = area reserved
                7E 01 90  0516  1439          MOVB    #DSC$K_CLASS_S,-(SP)     ; Class = static
                7E 0E 90  0519  1440          MOVB    #DSC$K_DTYPE_T,-(SP)     ; Data type = ASCII text
                7E 19 B0  051C  1441          MOVW    #INT$K_I_FRACT_D+7,-(SP) ; Length of string
                          051F  1442 ;+
                          051F  1443 ; Now call the conversion routine.
                          051F  1444 ;-
                08 AE 9F  051F  1445          PUSHAB  8(SP)                    ; Address of value
                04 AE 9F  0522  1446          PUSHAB  4(SP)                    ; Address of descriptor
       00000000'GF 02 FB  0525  1447          CALLS   #2,G^OTS$CVT_T_D         ; Convert the number
                0B 50 E9  052C  1448          BLBC    R0,2$                    ; Failure -- must be overflow
             67 08 AE 70  052F  1449          MOVD    8(SP),(R7)               ; Store result for user
             50 01 D0     0533  1450          MOVL    #1,R0                    ; Indicate success
             5E 29 C0     0536  1451          ADDL2   #INT$K_I_FRACT_D+7+8+8,SP ; Delete stack temps
                05        0539  1452          RSB                              ; Return to caller
                          053A  1453 ;+
                          053A  1454 ; Come here if the conversion fails.  This means that the intermediate-
                          053A  1455 ; form number cannot be represented as a double-precision floating point
                          053A  1456 ; number.  Store the reserved operand and return a failure indication.
                          053A  1457 ;-
             5E 29 C0     053A  1458 2$:        ADDL2   #INT$K_I_FRACT_D+7+8+8,SP ; Delete stack temps
          67 01 0F 79     053D  1459 1$:        ASHQ    #15,#1,(R7)              ; Store floating reserved operand
                50 D4     0541  1460          CLRL    R0                       ; Indicate failure
                05        0543  1461          RSB                              ; Return to caller
                          0544  1462 ;+
                          0544  1463 ; Come here if the exponent overflowed 2 decimal digits.  The
                          0544  1464 ; intermediate-form number is messed up.  Signal an internal OTS
                          0544  1465 ; failure.
                          0544  1466 ;-
        00000000'8F DD    0544  1467 3$:        PUSHL   #OTS$_FATINTERR          ; OTS internal failure
00000000'GF 01 FB         054A  1468          CALLS   #1,G^LIB$STOP
                          0551  1469
                          0551  1470 ;+
                          0551  1471 ; Out-of-range exponenent encountered for a CIT. Complain about it.
                          0551  1472 ;-
                          0551  1473 4$:
        00000000'8F DD    0551  1474          PUSHL   #COB$_INTRESOPE          ; CIT reserved operand signal
00000000'GF 01 FB         0557  1475          CALLS   #1, G^LIB$STOP           ; signal and stop.
```

```
                                055E  1477          .SBTTL  COB$CVTIP_R9    Convert Intermediate to Packed
                                055E  1478
                                055E  1479 ;++
                                055E  1480 ; FUNCTIONAL DESCRIPTION:
                                055E  1481 ;
                                055E  1482 ;       Convert an intermediate number to packed integer.
                                055E  1483 ;
                                055E  1484 ; CALLING SEQUENCE:
                                055E  1485 ;
                                055E  1486 ;       JSB COB$CVTIP_R9 (scale.rl.v, src.rx.r, dstlen.rl.v, dst.wp.r)
                                055E  1487 ;
                                055E  1488 ;       Arguments are passed in R6, R7, R8 and R9.
                                055E  1489 ;
                                055E  1490 ; INPUT PARAMETERS:
                                055E  1491 ;
                                055E  1492 ;       SCALE.rl.v              The power of ten by which the internal
                                055E  1493 ;                               representation of the source must be
                                055E  1494 ;                               multiplied to scale the same as the
                                055E  1495 ;                               internal representation of the dest.
                                055E  1496 ;       SRC.rx.r                The number to be converted
                                055E  1497 ;       DSTLEN.rl.v             The number of digits in the destination
                                055E  1498 ;
                                055E  1499 ; IMPLICIT INPUTS:
                                055E  1500 ;
                                055E  1501 ;       All of the trap bits in the PSL are assumed off.
                                055E  1502 ;
                                055E  1503 ; OUTPUT PARAMETERS:
                                055E  1504 ;
                                055E  1505 ;       DST.wp.r                The place to store the converted number
                                055E  1506 ;
                                055E  1507 ; IMPLICIT OUTPUTS:
                                055E  1508 ;
                                055E  1509 ;       NONE
                                055E  1510 ;
                                055E  1511 ; FUNCTION VALUE:
                                055E  1512 ;
                                055E  1513 ;       1 = SUCCESS, 0 = FAILURE
                                055E  1514 ;
                                055E  1515 ; SIDE EFFECTS:
                                055E  1516 ;
                                055E  1517 ;       Destroys registers R0 through R9.
                                055E  1518 ;
                                055E  1519 ;--
                                055E  1520
                                055E  1521 COB$CVTIP_R9::
                0063 8F   67    B1  055E  1522          CMPW    INT$W_I_EXP(R7), #INT$K_I_EXP_HI ; Bigger than biggest ?
                          2B    14  0563  1523          BGTR    12$                              ; Yes, overflow
                FF9D 8F   67    B1  0565  1524          CMPW    INT$W_I_EXP(R7), #INT$K_I_EXP_LO ; Smaller than smallest
                          24    19  056A  1525          BLSS    12$                              ; Yes, underflow
                      02  A7    95  056C  1526          TSTB    INT$P_I_FRACT(R7)        ; Is source zero?
                          2F    13  056F  1527          BEQL    13$                      ; Yes, return zero.
                      56  67    A0  0571  1528          ADDW2   INT$W_I_EXP(R7),R6       ; Compute adjusted scale
                      56  12    A2  0574  1529          SUBW2   #INT$K_I_FRACT_D,R6
                      50  56    33  0577  1530          CVTWB   R6,R0                    ; Shift count in signed byte range?
                          0D    1D  057A  1531          BVS     10$                      ; No, special handling
69    58    00    02  A7    12  56    F8  057C  1532          ASHP    R6,#INT$K_I_FRACT_D,INT$P_I_FRACT(R7),#0,R8,(R9)
                                0584  1533                                           ; Scale and integerize fraction
```

```
                     0584  1534                                ;       (also clears R0)
                07   1D   0584  1535          BVS     11$     ; Overflow means can't convert
                50   D6   0586  1536          INCL    R0      ; Indicate success, R0 = 1
                     05   0588  1537          RSB             ; Return to caller.
                     0589  1538 ;+
                     0589  1539 ; Come here if the shift count is not in signed byte range.  If it is
                     0589  1540 ; positive, an overflow exists.  If negative, return zero.
                     0589  1541 ;-
             13 56   0F   E0   0589  1542 10$:    BBS     #15,R6,13$      ; Branch if negative
                     058D  1543 ;+
                     058D  1544 ; Come here if the number cannot be represented in the number of digits
                     058D  1545 ; provided.
                     058D  1546 ;-
                50   D4   058D  1547 11$:    CLRL    R0              ; Indicate failure
                     05   058F  1548          RSB                     ; Return to caller.
                     0590  1549 ;+
                     0590  1550 ; Come here on reserved operand.  Signal the error and give the
                     0590  1551 ; failure return.
                     0590  1552 ;-
      00000000'8F    DD   0590  1553 12$:    PUSHL   #COB$_INTRESOPE         ; Intermediate reserved operand
00000000'GF    01    FB   0596  1554          CALLS   #1,G^LIB$SIGNAL         ; Print message for user
                50   D4   059D  1555          CLRL    R0                      ; Indicate failure
                     05   059F  1556          RSB                             ; Return to caller.
                     05A0  1557 ;+
                     05A0  1558 ; Come here if the source is zero or the shift count is too small for
                     05A0  1559 ; the ASHP instruction.  Return a zero.
                     05A0  1560 ;-
          69 58 00   F9   05A0  1561 13$:    CVTLP   #0,R8,(R9)              ; Set result to zero
             50 01   D0   05A4  1562          MOVL    #1,R0                   ; Indicate success
                     05   05A7  1563          RSB                             ; Return to caller.
```

```
                     05A8  1565          .SBTTL  COB$CVTRIP_R9   Convert Rounded Intermediate to Packed
                     05A8  1566
                     05A8  1567  ;++
                     05A8  1568  ; FUNCTIONAL DESCRIPTION:
                     05A8  1569  ;
                     05A8  1570  ;       Convert an intermediate number to packed integer.
                     05A8  1571  ;
                     05A8  1572  ; CALLING SEQUENCE:
                     05A8  1573  ;
                     05A8  1574  ;       JSB COB$CVTRIP_R9 (scale.rl.v, src.rx.r, dstlen.rl.v, dst.wp.r)
                     05A8  1575  ;
                     05A8  1576  ;       Arguments are passed in R6, R7, R8 and R9.
                     05A8  1577  ;
                     05A8  1578  ; INPUT PARAMETERS:
                     05A8  1579  ;
                     05A8  1580  ;       SCALE.rl.v              The power of ten by which the internal
                     05A8  1581  ;                               representation of the source must be
                     05A8  1582  ;                               multiplied to scale the same as the
                     05A8  1583  ;                               internal representation of the dest.
                     05A8  1584  ;       SRC.rx.r                The number to be converted
                     05A8  1585  ;       DSTLEN.rl.v             The number of digits in the destination
                     05A8  1586  ;
                     05A8  1587  ; IMPLICIT INPUTS:
                     05A8  1588  ;
                     05A8  1589  ;       All of the trap bits in the PSL are assumed off.
                     05A8  1590  ;
                     05A8  1591  ; OUTPUT PARAMETERS:
                     05A8  1592  ;
                     05A8  1593  ;       DST.wp.r                The place to store the converted number
                     05A8  1594  ;
                     05A8  1595  ; IMPLICIT OUTPUTS:
                     05A8  1596  ;
                     05A8  1597  ;       NONE
                     05A8  1598  ;
                     05A8  1599  ; FUNCTION VALUE:
                     05A8  1600  ;
                     05A8  1601  ;       1 = SUCCESS, 0 = FAILURE
                     05A8  1602  ;
                     05A8  1603  ; SIDE EFFECTS:
                     05A8  1604  ;
                     05A8  1605  ;       Destroys registers R0 through R9.
                     05A8  1606  ;
                     05A8  1607  ;--
                     05A8  1608
                     05A8  1609  COB$CVTRIP_R9::
           0063 8F    67   B1   05A8  1610          CMPW    INT$W_I_EXP(R7), #INT$K_I_EXP_HI ; Bigger than biggest ?
                      2B   14   05AD  1611          BGTR    12$                               ; Yes, overflow
         FF9D 8F      67   B1   05AF  1612          CMPW    INT$W_I_EXP(R7), #INT$K_I_EXP_LO ; Smaller than smallest
                      24   19   05B4  1613          BLSS    12$                               ; Yes, underflow
                02    A7   95   05B6  1614          TSTB    INT$P_I_FRACT(R7)         ; Is source zero?
                      2F   13   05B9  1615          BEQL    13$                       ; Yes, return zero.
                56    67   A0   05BB  1616          ADDW2   INT$W_I_EXP(R7),R6        ; Compute adjusted scale
                56    12   A2   05BE  1617          SUBW2   #INT$K_I_FRACT_D,R6
                50    56   33   05C1  1618          CVTWB   R6,R0                     ; Shift count in signed byte range?
                      0D   1D   05C4  1619          BVS     10$                       ; No, special handling
69   58   05   02 A7   12   56   F8   05C6  1620          ASHP    R6,#INT$K_I_FRACT_D,INT$P_I_FRACT(R7),#5,R8,(R9)
                                05CE  1621                                            ; Scale and integerize fraction
```

```
                05CE  1622                                      ;       (also clears R0)
          07    1D    05CE  1623          BVS     11$           ; Overflow means can't convert
          50    D6    05D0  1624          INCL    R0            ; Indicate success, R0 = 1
          05          05D2  1625          RSB                   ; Return to caller.
                      05D3  1626 ;+
                      05D3  1627 ; Come here if the shift count is not in signed byte range.  If it is
                      05D3  1628 ; positive, an overflow exists.  If negative, return zero.
                      05D3  1629 ;-
    13 56     0F    E0    05D3  1630 10$:    BBS     #15,R6,13$    ; Branch if negative
                      05D7  1631 ;+
                      05D7  1632 ; Come here if the number cannot be represented in the number of digits
                      05D7  1633 ; provided.
                      05D7  1634 ;-
          50    D4    05D7  1635 11$:    CLRL    R0            ; Indicate failure
          05          05D9  1636          RSB                   ; Return to caller.
                      05DA  1637 ;+
                      05DA  1638 ; Come here on reserved operand.  Signal the error and give the failure
                      05DA  1639 ; return to the user.
                      05DA  1640 ;-
  00000000'8F     DD    05DA  1641 12$:    PUSHL   #COB$_INTRESOPE ; Intermediate reserved operand
00000000'GF       01    FB    05E0  1642          CALLS   #1,G^LIB$SIGNAL ; Print message for user
          50    D4    05E7  1643          CLRL    R0            ; Indicate failure
          05          05E9  1644          RSB                   ; Return to caller.
                      05EA  1645 ;+
                      05EA  1646 ; Come here if the source is zero or if the shift count is too small
                      05EA  1647 ; for the ASHP instruction.  Return a zero.
                      05EA  1648 ;-
    69    58    00    F9    05EA  1649 13$:    CVTLP   #0,R8,(R9)    ; Set result to zero
          50    01    D0    05EE  1650          MOVL    #1,R0         ; Indicate success
                      05F1  1651          RSB                   ; Return to caller.
```

```
                     05F2  1653          .SBTTL  COB$CVTTI_R8    Convert Text to Intermediate
                     05F2  1654
                     05F2  1655 ;++
                     05F2  1656 ; FUNCTIONAL DESCRIPTION:
                     05F2  1657 ;
                     05F2  1658 ;       Convert text to intermediate
                     05F2  1659 ;
                     05F2  1660 ; CALLING SEQUENCE:
                     05F2  1661 ;
                     05F2  1662 ;       JSB COB$CVTTI_R8 (srclen.rl.v, src.rt.r, dst.wx.r)
                     05F2  1663 ;
                     05F2  1664 ;       Arguments are passed in R6, R7 and R8
                     05F2  1665 ;
                     05F2  1666 ; INPUT PARAMETERS:
                     05F2  1667 ;
                     05F2  1668 ;       SRCLEN.rl.v             The number of digits in the source
                     05F2  1669 ;       SRC.rt.r                The number to be converted
                     05F2  1670 ;
                     05F2  1671 ; IMPLICIT INPUTS:
                     05F2  1672 ;
                     05F2  1673 ;       All of the trap bits in the PSL are assumed off.
                     05F2  1674 ;
                     05F2  1675 ; OUTPUT PARAMETERS:
                     05F2  1676 ;
                     05F2  1677 ;       DST.wx.r                The place to store the converted number
                     05F2  1678 ;
                     05F2  1679 ; IMPLICIT OUTPUTS:
                     05F2  1680 ;
                     05F2  1681 ;       NONE
                     05F2  1682 ;
                     05F2  1683 ; FUNCTION VALUE:
                     05F2  1684 ;
                     05F2  1685 ;       1 = SUCCESS, 0 = FAILURE
                     05F2  1686 ;
                     05F2  1687 ; SIDE EFFECTS:
                     05F2  1688 ;
                     05F2  1689 ;       Destroys registers R0 through R8.
                     05F2  1690 ;
                     05F2  1691 ;--
                     05F2  1692
                     05F2  1693 COB$CVTTI_R8::
          5E   13    C2  05F2  1694          SUBL2   #INT$K_I_FRACT_D+1,SP   ; Space for separate-sign fraction
                     05F5  1695 ;+
                     05F5  1696 ; Since the compiler will convert the text string with in-line code if it can,
                     05F5  1697 ; there is no need to include a test for this case.  Thus we will assume that
                     05F5  1698 ; the string is too long to be converted using decimal string instructions.
                     05F5  1699 ;-
    02 A8  12   00   F9  05F5  1700          CVTLP   #0,#INT$K_I_FRACT_D,INT$P_I_FRACT(R8)
                     05FA  1701                                          ; Initialize result to zero
          68   B4    05FA  1702          CLRW    INT$W_I_EXP(R8)         ; Also set exponent to zero
                     05FC  1703 ;+
                     05FC  1704 ; Delete leading zeros from the source string so that the result will
                     05FC  1705 ; contain the maximum number of significant digits and be normalized.
                     05FC  1706 ;-
          56   D5    05FC  1707          TSTL    R6                      ; Any digits?
          45   15    05FE  1708          BLEQ    14$                     ; No, return the zero.
    30    87   91    0600  1709 10$:     CMPB    (R7)+,#^A/0/            ; Is this digit zero?
```

```
                     05   12   0603  1710          BNEQ    11$                             ; No, start collecting digits
                  F8 56   F5   0605  1711          SOBGTR  R6,10$                          ; Loop back for another
                     3B   11   0608  1712          BRB     14$                             ; All digits zero, return zero.
                                060A  1713 ;+
                                060A  1714 ; Come here at the first non-zero digit of the string.  At this time, R6
                                060A  1715 ; contains the number of digits remaining in the string, which is the
                                060A  1716 ; exponent of the intermediate-form number.  We can now store it and test the
                                060A  1717 ; exponent for overflow.
                                060A  1718 ;-
           00000063 8F          56   D1  060A  1719 11$:     CMPL    R6,#INT$K_I_EXP_HI      ; Is exponent too big?
                                55   14   0611  1720          BGTR    16$                     ; Yes, fail to convert.
                           68   56   B0   0613  1721          MOVW    R6,INT$W_I_EXP(R8)      ; Store exponent
                                57   D7   0616  1722          DECL    R7                      ; Correct for autoincrement
                                0618  1723 ;+
                                0618  1724 ; Build a leading separate sign representation of the first 18 digits.
                                0618  1725 ;-
    01 AE    12   30   6E   00   2C  0618  1726          MOVC5   #0,(SP),#^A/0/,#INT$K_I_FRACT_D,1(SP)
                                061F  1727                                                  ; Zero-fill the digits
                           6E   2B   90  061F  1728          MOVB    #^A/+/,(SP)             ; Insert plus sign
                                52   D4   0622  1729          CLRL    R2                      ; Initialize digit counter
                           50   87   9A  0624  1730 12$:     MOVZBL  (R7)+,R0                ; Get digit from input
                      51   50   30   C3  0627  1731          SUBL3   #^A/0/,R0,R1            ; Convert to binary
                           09   51   D1  062B  1732          CMPL    R1,#9                   ; Check range
                                1C   1A  062E  1733          BGTRU   15$                     ; Br if not a digit
                           12   52   D1  0630  1734          CMPL    R2,#18                  ; Do not store more than 18 digits
                                07   1E  0633  1735          BGEQU   13$                     ;
                      01 AE42   50   90  0635  1736          MOVB    R0,1(SP)[R2]            ; Store the digit
                                52   D6  063A  1737          INCL    R2                      ; Count the digit
                         E5 56   F5   063C  1738 13$:     SOBGTR  R6,12$                  ; Loop if more digits
                                063F  1739 ;+
                                063F  1740 ; Come here when we have processed all digits.
                                063F  1741 ;-
02 A8    12   6E   12   09  063F  1742          CVTSP   #INT$K_I_FRACT_D,(SP),#INT$K_I_FRACT_D,INT$P_I_FRACT(R8)
                                0645  1743                                                  ; Get packed fraction
                           5E   13   C0  0645  1744 14$:     ADDL2   #INT$K_I_FRACT_D+1,SP   ; Delete stack temps
                           50   01   D0  0648  1745          MOVL    #1,R0                   ; Indicate success
                                05   064B  1746          RSB                             ; Return
                                064C  1747 ;+
                                064C  1748 ; Come here on an invalid digit.  Signal the error and indicate failure.
                                064C  1749 ; Return the reserved operand.  The invalid digit is in R0.
                                064C  1750 ;-
                           01   DD   064C  1751 15$:     PUSHL   #1                      ; Construct ASCIC digit at top
                      01 AE   50   90   064E  1752          MOVB    R0,1(SP)                ;     of stack
                                0652  1753                                                  ; This is the FAO argument
                           5E   DD   0652  1754          PUSHL   SP                      ; Address of FAO argument
                           01   DD   0654  1755          PUSHL   #1                      ; Count of FAO arguments
                 00000000'8F   DD   0656  1756          PUSHL   #COB$_INVDECDIG         ; Message code: Invalid decimal digit
              00000000'GF      03   FB   065C  1757          CALLS   #3,G^LIB$SIGNAL         ; Print message for user
                           5E   04   C0   0663  1758          ADDL2   #4,SP                   ; Remove FAO argument
                                0D   11   0666  1759          BRB     17$                     ; Set reserved operand and return
                                0668  1760 ;+
                                0668  1761 ; Come here if there are so many digits that the exponent overflows.
                                0668  1762 ; Store the reserved operand, signal the error and give the failure
                                0668  1763 ; return.
                                0668  1764 ;-
                 00000000'8F   DD   0668  1765 16$:     PUSHL   #COB$_INTEXPOVE         ; Intermediate exponent overflow
              00000000'GF      01   FB   066E  1766          CALLS   #1,G^LIB$SIGNAL         ; Print message for user
```

```
                68    8000 8F    B0   0675  1767 17$:    MOVW    #INT$K_I_EXP_RES,INT$W_I_EXP(R8)
                                    067A  1768                                        ; Exponent marks reserved operand
                           50    D4   067A  1769          CLRL    R0                 ; Indicate failure
                      5E   13    C0   067C  1770          ADDL2   #INT$K_I_FRACT_D+1,SP ; Delete stack temps
                           05    067F  1771              RSB                        ; Return
                                 0680  1772 ;
                                 0680  1773              .END                       ; End of module COB$INTER
```

```
BIAS                            00000000 R      02
BIAS_1                          00000006 R      02
BIAS_DIGITS                   = 0000000A
COB$CNVOUT                      ********    X   00
COB$CVTDI_R7                    0000014F RG     02
COB$CVTFI_R7                    000000FF RG     02
COB$CVTID_R7                    000004DE RG     02
COB$CVTIF_R7                    0000045C RG     02
COB$CVTIL_R8                    000002AA RG     02
COB$CVTIP_R9                    0000055E RG     02
COB$CVTIQ_R8                    0000035C RG     02
COB$CVTIW_R8                    000001F2 RG     02
COB$CVTLI_R8                    0000008C RG     02
COB$CVTPI_R9                    0000019F RG     02
COB$CVTQI_R8                    0000009B RG     02
COB$CVTRIC_R8                   00000303 RG     02
COB$CVTRIP_R9                   000005A8 RG     02
COB$CVTRIQ_R8                   000003DC RG     02
COB$CVTRIW_R8                   0000024E RG     02
COB$CVTTI_R8                    000005F2 RG     02
COB$CVTWI_R8                    0000007A RG     02
COB$_INTEXPOVE                  ********    X   00
COB$_INTEXPUND                  ********    X   00
COB$_INTRESOPE                  ********    X   00
COB$_INVDECDIG                  ********    X   00
DSC$K_CLASS_S                 = 00000001
DSC$K_DTYPE_T                 = 0000000E
INT$$NORMALIZE                  0000000C R      02
INT$K_I_EXP_HI                = 00000063
INT$K_I_EXP_LO                = FFFFFF9D
INT$K_I_EXP_RES               = FFFF8000
INT$K_I_FRACT_D               = 00000012
INT$K_I_FRACT_L               = 0000000A
INT$K_I_LEN                   = 0000000C
INT$P_I_FRACT                 = 00000002
INT$W_I_EXP                   = 00000000
LIB$SIGNAL                      ********    X   00
LIB$STOP                        ********    X   00
OTS$CVT_T_D                     ********    X   00
OTS$_FATINTERR                  ********    X   00
```

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

```
PSECT name                      Allocation          PSECT No.  Attributes
----------                      ----------          ---------  ----------
.  ABS  .                       00000000  (    0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD  NOWRT NOVEC BYTE
$ABS$                           00000000  (    0.)  01 (  1.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD    WRT NOVEC BYTE
_COB$CODE                       00000680  ( 1664.)  02 (  2.)    PIC   USR   CON   REL   LCL   SHR   EXE   RD  NOWRT NOVEC LONG
```

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	31	00:00:00.11	00:00:01.20
Command processing	114	00:00:00.37	00:00:03.37
Pass 1	207	00:00:03.37	00:00:19.52
Symbol table sort	0	00:00:00.14	00:00:00.79
Pass 2	301	00:00:02.26	00:00:11.64
Symbol table output	7	00:00:00.04	00:00:00.04
Psect synopsis output	4	00:00:00.02	00:00:00.04
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	666	00:00:06.31	00:00:36.60

The working set limit was 1650 pages.
33510 bytes (66 pages) of virtual memory were used to buffer the intermediate code.
There were 20 pages of symbol table space allocated to hold 172 non-local and 68 local symbols.
1773 source lines were read in Pass 1, producing 15 object records in Pass 2.
9 pages of virtual memory were used to define 8 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$:COBINTER/OBJ=OBJ$:COBINTER MSRC$:COBINTER/UPDATE=(ENH$:COBINTER)+LIB$:C
```

COBPAUSE
LIS

COBMSG
LIS

COBHANDLE
LIS

COBINTAR
LIS

COBINTER
LIS

COBMULQ
LIS

COBPOSERA
LIS

COBIOEXCE
LIS

COBLINAGE
LIS

COBKEY
LIS

COBINVUSE
LIS