```
BBBBBBBBBBBB        AAAAAAAAA       SSSSSSSSSSS  RRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
BBBBBBBBBBBB        AAAAAAAAA       SSSSSSSSSSS  RRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
BBBBBBBBBBBB        AAAAAAAAA       SSSSSSSSSSS  RRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
BBB          BBB   AAA         AAA  SSS          RRR         RRR       TTT        LLL
BBB          BBB   AAA         AAA  SSS          RRR         RRR       TTT        LLL
BBB          BBB   AAA         AAA  SSS          RRR         RRR       TTT        LLL
BBB          BBB   AAA         AAA  SSS          RRR         RRR       TTT        LLL
BBB          BBB   AAA         AAA  SSS          RRR         RRR       TTT        LLL
BBB          BBB   AAA         AAA  SSS          RRR         RRR       TTT        LLL
BBBBBBBBBBBB       AAA         AAA     SSSSSSSSS    RRRRRRRRRRR       TTT        LLL
BBBBBBBBBBBB       AAA         AAA     SSSSSSSSS    RRRRRRRRRRR       TTT        LLL
BBBBBBBBBBBB       AAA         AAA     SSSSSSSSS    RRRRRRRRRRR       TTT        LLL
BBB          BBB   AAAAAAAAAAAAAAA           SSS  RRR   RRR         TTT        LLL
BBB          BBB   AAAAAAAAAAAAAAA           SSS  RRR   RRR         TTT        LLL
BBB          BBB   AAAAAAAAAAAAAAA           SSS  RRR   RRR         TTT        LLL
BBB          BBB   AAA         AAA           SSS  RRR      RRR      TTT        LLL
BBB          BBB   AAA         AAA           SSS  RRR      RRR      TTT        LLL
BBB          BBB   AAA         AAA           SSS  RRR      RRR      TTT        LLL
BBBBBBBBBBBB       AAA         AAA  SSSSSSSSSSS  RRR         RRR    TTT        LLLLLLLLLLLLLLL
BBBBBBBBBBBB       AAA         AAA  SSSSSSSSSSS  RRR         RRR    TTT        LLLLLLLLLLLLLLL
BBBBBBBBBBBB       AAA         AAA  SSSSSSSSSSS  RRR         RRR    TTT        LLLLLLLLLLLLLLL
```

```
BBBBBBBB      AAAAAA      SSSSSSSS  EEEEEEEEEE  DDDDDDDD      IIIIII    VV      VV  PPPPPPPP
BBBBBBBB      AAAAAA      SSSSSSSS  EEEEEEEEEE  DDDDDDDD      IIIIII    VV      VV  PPPPPPPP
BB      BB   AA    AA   SS        EE          DD      DD      II      VV      VV  PP      PP
BB      BB   AA    AA   SS        EE          DD      DD      II      VV      VV  PP      PP
BB      BB   AA    AA   SS        EE          DD      DD      II      VV      VV  PP      PP
BB      BB   AA    AA   SS        EE          DD      DD      II      VV      VV  PP      PP
BBBBBBBB     AA    AA     SSSSSS  EEEEEEEE    DD      DD      II      VV      VV  PPPPPPPP
BBBBBBBB     AA    AA     SSSSSS  EEEEEEEE    DD      DD      II      VV      VV  PPPPPPPP
BB      BB   AAAAAAAAAA         SS  EE          DD      DD      II      VV      VV  PP
BB      BB   AAAAAAAAAA         SS  EE          DD      DD      II      VV      VV  PP
BB      BB   AA    AA           SS  EE          DD      DD      II        VV  VV    PP
BB      BB   AA    AA           SS  EE          DD      DD      II        VV  VV    PP
BBBBBBBB     AA    AA   SSSSSSSS  EEEEEEEEEE  DDDDDDDD      IIIIII      VV      PP
BBBBBBBB     AA    AA   SSSSSSSS  EEEEEEEEEE  DDDDDDDD      IIIIII      VV      PP

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
```

```
;     1          0001  0 MODULE BAS$EXTEND_DIVP (
;     2          0002  0                     IDENT = '1-005'         ! File: BASEDIVP.B32 EDIT:MDL1005
;     3          0003  0                     ) =
;     4          0004  1 BEGIN
;     5          0005  1 !
;     6          0006  1 !****************************************************************************
;     7          0007  1 !*                                                                          *
;     8          0008  1 !*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
;     9          0009  1 !*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
;    10          0010  1 !*  ALL RIGHTS RESERVED.                                                     *
;    11          0011  1 !*                                                                          *
;    12          0012  1 !*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
;    13          0013  1 !*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE  *
;    14          0014  1 !*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
;    15          0015  1 !*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
;    16          0016  1 !*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
;    17          0017  1 !*  TRANSFERRED.                                                             *
;    18          0018  1 !*                                                                          *
;    19          0019  1 !*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
;    20          0020  1 !*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
;    21          0021  1 !*  CORPORATION.                                                             *
;    22          0022  1 !*                                                                          *
;    23          0023  1 !*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
;    24          0024  1 !*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
;    25          0025  1 !*                                                                          *
;    26          0026  1 !*                                                                          *
;    27          0027  1 !****************************************************************************
;    28          0028  1 !
;    29          0029  1
;    30          0030  1 !++
;    31          0031  1 ! FACILITY:  BASIC
;    32          0032  1 !
;    33          0033  1 ! ABSTRACT:
;    34          0034  1 !
;    35          0035  1 !       This module performs double precision division for scaled decimal
;    36          0036  1 !       data types.
;    37          0037  1 !
;    38          0038  1 ! ENVIRONMENT:  VAX-11 User Mode
;    39          0039  1 !
;    40          0040  1 ! AUTHOR: Bob Hanek,     CREATION DATE: 20-Jan-1982
;    41          0041  1 !
;    42          0042  1 ! MODIFIED BY:
;    43          0043  1 !
;    44          0044  1 ! 1-001 - Original.  RNH 20-Jan-1982
;    45          0045  1 ! 1-002 - Change name to BAS$EXTEND_DIVP and change errors to
;    46          0046  1 !         Decimal Error.  PLL 12-Feb-1982
;    47          0047  1 ! 1-003 - Check the decimal overflow flag within the flags word found
;    48          0048  1 !         in the BASIC frame and use the value of that setting to
;    49          0049  1 !         set the PSW accordingly.  LB 15-May-1982
;    50          0050  1 ! 1-004 - Changed the declarations of A_EXP and B_EXP from BYTE to
;    51          0051  1 !         SIGNED BYTE so that negative values will not be zero
;    52          0052  1 !         extended from a byte to a longword for computational
;    53          0053  1 !         purposes.  Added the variable SAV_BADDR to save the value of
;    54          0054  1 !         B_ADDR, which was being altered by LIB$$UNPACK_SD_R8. Both the
;    55          0055  1 !         altered value and the old value are needed. JCW 29-Nov-1982
;    56          0056  1 ! 1-005 - RND_TRUNC should be figured into the shift factor when calling
;    57          0057  1 !         LIB$$CVT_A_AP_R8.  MDL 5-Oct-1983
```

```
;    58          0058  1 !--
;    59          0059  1
;    60          0060  1 !<BLF/PAGE>
```

```
;    62          0061  1 !+
;    63          0062  1 ! SWITCHES:
;    64          0063  1 !-
;    65          0064  1
;    66          0065  1 SWITCHES ADDRESSING_MODE (EXTERNAL = GENERAL, NONEXTERNAL = WORD_RELATIVE);
;    67          0066  1
;    68          0067  1 !+
;    69          0068  1 ! LINKAGES:
;    70          0069  1 !-
;    71          0070  1
;    72          0071  1 LINKAGE
;    73          0072  1         JSB1 = JSB (REGISTER=6, REGISTER=7) : NOPRESERVE (2,3,4,5),
;    74          0073  1         JSB2 = JSB (REGISTER=6, REGISTER=7, REGISTER=8) : NOPRESERVE (2,3,4,5),
;    75          0074  1         JSB3 = JSB (REGISTER=6, REGISTER=7, REGISTER=8, REGISTER=9)
;    76          0075  1                 : NOPRESERVE (2,3,4,5),
;    77          0076  1         JSB4 = JSB (REGISTER=6, REGISTER=7, REGISTER=8, REGISTER=9, REGISTER=10)
;    78          0077  1                 : NOPRESERVE (2,3,4,5),
;    79          0078  1         JSB5 = JSB (REGISTER=4, REGISTER=5, REGISTER=6, REGISTER=7, REGISTER=8)
;    80          0079  1                 : NOPRESERVE (2,3),
;    81          0080  1         JSB6 = JSB (REGISTER=3, REGISTER=4, REGISTER=5, REGISTER=6, REGISTER=7,
;    82          0081  1                   REGISTER=8) : NOPRESERVE (2,3);
;    83          0082  1
;    84          0083  1 !+
;    85          0084  1 ! TABLE OF CONTENTS:
;    86          0085  1 !-
;    87          0086  1
;    88          0087  1 FORWARD ROUTINE
;    89          0088  1         BAS$EXTEND_DIVP : NOVALUE;      ! Double precision packed decimal division
;    90          0089  1
;    91          0090  1 !+
;    92          0091  1 ! INCLUDE FILES:
;    93          0092  1 !-
;    94          0093  1
;    95          0094  1 REQUIRE 'RTLIN:RTLPSECT';                       ! Macros for defining psects
;    96          0189  1 REQUIRE 'RTLIN:BASFRAME';                       ! Define frame offsets
;    97          0392  1 LIBRARY 'RTLSTARLE';                            ! System definitions
;    98          0393  1
;    99          0394  1 !+
;   100          0395  1 ! MACROS:
;   101          0396  1 !
;   102          0397  1 !       NONE
;   103          0398  1 !-
;   104          0399  1
;   105          0400  1 !+
;   106          0401  1 ! PSECTS:
;   107          0402  1 !-
;   108          0403  1
;   109          0404  1 DECLARE_PSECTS (BAS);                           ! Declare psects for BAS$ facility
;   110          0405  1
;   111          0406  1 !+
;   112          0407  1 ! OWN STORAGE:
;   113          0408  1 !
;   114          0409  1 !       NONE
;   115          0410  1 !-
;   116          0411  1
;   117          0412  1 !+
;   118          0413  1 ! EXTERNAL REFERENCES:
```

```
;  119          0414  1 !-
;  120          0415  1
;  121          0416  1 EXTERNAL ROUTINE
;  122          0417  1         BAS$$STOP,                          ! Signal fatal error
;  123          0418  1         LIB$$CALC_D_R7:JSB1,                ! Calculates normalization factor
;  124          0419  1         LIB$$CALC_Q_R9:JSB3 NOVALUE,        ! Calculates one quotient digit
;  125          0420  1         LIB$$SUB_PACK_R8:JSB2,              ! Subtracts two decimal arrays
;  126          0421  1         LIB$$CVT_AP_P_R8:JSB6,              ! Converts an packed array to a single
;  127          0422  1                                             !    rounded packed string
;  128          0423  1         LIB$$UNPACK_SD_R8:JSB5 NOVALUE,     ! Unpacks SD decscipter, convert packed
;  129          0424  1                                             !    decimal string to leading separate
;  130          0425  1         LIB$$MUL_PACK_R10:JSB4 NOVALUE,     ! Multiplies a packed array by a single
;  131          0426  1                                             !    entry
;  132          0427  1         LIB$$ADJUST_Q_R9:JSB3 NOVALUE,      ! Adjusts intermediate results of divi-
;  133          0428  1                                             !    sion algorithm if initial quess at
;  134          0429  1                                             !    a quotient digit is wrong
;  135          0430  1         LIB$$CVT_STR_PACK_R9:JSB3 NOVALUE,
;  136          0431  1                                             ! Converts a string of decimal digits
;  137          0432  1                                             !    to an array of packed decimal
;  138          0433  1                                             !    values
;  139          0434  1         BAS$HANDLER;
;  140          0435  1
;  141          0436  1 BIND
;  142          0437  1         ZERO    = UPLIT BYTE (REP 15 OF (%X'00'),%X'0C'); ! Packed zero
;  143          0438  1
;  144          0439  1 BUILTIN
;  145          0440  1         BICPSW,                             ! Bit clear PSW
;  146          0441  1         BISPSW,                             ! Bit set PSW
;  147          0442  1         MOVP;                               ! Move packed decimal data
;  148          0443  1
;  149          0444  1 !+
;  150          0445  1 ! The following are the error codes produced by this module.
;  151          0446  1 !-
;  152          0447  1
;  153          0448  1 EXTERNAL LITERAL
;  154          0449  1         BAS$K_DECERR,                       ! Decimal overflow
;  155          0450  1         BAS$K_DIVBY_ZER;                    ! Divide by zero
;  156          0451  1
;  157          0452  1
```

```
;  159          0453  1 GLOBAL ROUTINE BAS$EXTEND_DIVP (
;  160          0454  1                 A,                      ! Address of descripter for A
;  161          0455  1                 B,                      ! Address of descripter for B
;  162          0456  1                 C,                      ! Address of descripter for C
;  163          0457  1                 RND_TRUNC):NOVALUE =    ! Round/Truncate parameter
;  164          0458  1
;  165          0459  1 !++
;  166          0460  1 !
;  167          0461  1 ! FUNCTIONAL DESCRIPTION:
;  168          0462  1 !
;  169          0463  1 !  This routine finds the quotient of two scaled packed decimal strings i.e.
;  170          0464  1 !  C = A / B.  The algorithm implemented here has been provided by KNUTH.  It
;  171          0465  1 !  is his famous Algorithm D (division of non-negative integers (which has
;  172          0466  1 !  been modified to handle negative integers)) found in Volume 2 of
;  173          0467  1 !  that extraordinary series (Vol. 2 is entitled Seminumerical Algorithms).
;  174          0468  1 !  An explanation of the algorithm appears further on in the program.
;  175          0469  1 !
;  176          0470  1 ! CALLING SEQUENCE:
;  177          0471  1 !
;  178          0472  1 !       BAS$EXTEND_DIVP (A.rp.dsd, B.rp.dsd, C.rp.dsd, RND_TRUNC.rb.v)
;  179          0473  1 !
;  180          0474  1 ! FORMAL PARAMETERS:
;  181          0475  1 !
;  182          0476  1 !       A.rp.dsd        Descripter for A
;  183          0477  1 !       B.rp.dsd        Descripter for B
;  184          0478  1 !       C.rp.dsd        Descripter for C
;  185          0479  1 !       RND_TRUNC.rb.v  Round/Truncate parameter (0 = truncate, 5 = round)
;  186          0480  1 !
;  187          0481  1 ! IMPLICIT INPUTS:
;  188          0482  1 !
;  189          0483  1 !       -NONE
;  190          0484  1 !
;  191          0485  1 ! IMPLICIT OUTPUTS:
;  192          0486  1 !
;  193          0487  1 !       -NONE
;  194          0488  1 !
;  195          0489  1 ! ROUTINE VALUE:
;  196          0490  1 !
;  197          0491  1 !       -NONE
;  198          0492  1 !
;  199          0493  1 ! COMPLETION CODES
;  200          0494  1 !
;  201          0495  1 !       -NONE
;  202          0496  1 !
;  203          0497  1 ! MACROS:
;  204          0498  1 !
;  205          0499  1 !       -NONE
;  206          0500  1 !
;  207          0501  1 ! SIDE EFFECTS:
;  208          0502  1 !
;  209          0503  1 !       -NONE
;  210          0504  1 !
;  211          0505  1 !-
```

```
;  213          0506  2 BEGIN
;  214          0507  2
;  215          0508  2     MAP
;  216          0509  2         C:REF BLOCK [12,BYTE];
;  217          0510  2
;  218          0511  2     STACKLOCAL
;  219          0512  2
;  220          0513  2         SAV_BADDR,                  ! Used to save the old value of B_ADDR
;  221          0514  2         A_LENGTH:WORD,              ! Number of digits in A string
;  222          0515  2         A_ADDR,                     ! Address of A string
;  223          0516  2         A_EXP:SIGNED BYTE,          ! Scale factor of A
;  224          0517  2         A_SIGN:BYTE,                ! Sign of A (0 for pos, 1 for neg)
;  225          0518  2         B_LENGTH:WORD,              ! Number of digits in B string
;  226          0519  2         B_ADDR,                     ! Address of B string
;  227          0520  2         B_EXP:SIGNED BYTE,          ! Scale factor of B
;  228          0521  2         B_SIGN:BYTE,                ! Sign of B (0 for pos, 1 for neg)
;  229          0522  2         C_SIGN:BYTE,                ! Sign of result digits
;  230          0523  2         A_LEN,                      ! Number of digits needed in A to produce result
;  231          0524  2         A_CHUNKS,                   ! Number of 15 digit chunks needed in A
;  232          0525  2         B_CHUNKS,                   ! Number of 15 digit chunks in B
;  233          0526  2         Q_LENGTH,                   ! Number of digit required in the quotient
;  234          0527  2         Q_CHUNKS,                   ! Number of 15 digit chunks required in quotient
;  235          0528  2         QBUF:VECTOR[128, BYTE],     ! Addr of 1st 15 digit chunks of the quotient
;  236          0529  2         STATUS,                     ! Longword for returning status
;  237          0530  2         FLAG,                       ! B_CHUNKS = 1 ==> FLAG = 1, FLAG = 0 otherwise
;  238          0531  2         FMP:REF BLOCK[0,BYTE] FIELD(BSF$FCD),    ! Pointer to FCD
;  239          0532  2         SAVE_FP:REF BLOCK[0,BYTE] FIELD(BSF$FCD);
;  240          0533  2
;  241          0534  2 BUILTIN
;  242          0535  2         FP;                         ! Frame Pointer
;  243          0536  2
;  244          0537  2 BIND
;  245          0538  2         ABUF    = QBUF[8],          ! Address of A buffer
;  246          0539  2         ABUF8   = QBUF[16],         ! Address of A buffer + 8 bytes
;  247          0540  2         BBUF    = QBUF[72],         ! Address of B buffer
;  248          0541  2         QBBUF   = QBUF[96],         ! Address of Q*B buffer
;  249          0542  2         QBBUF8  = QBUF[104];        ! Address of Q*B buffer + 8 bytes
;  250          0543  2
```

```
;  252          0544  2 !**********************************************************************
;  253          0545  2 !                                                                     *
;  254          0546  2 !                          THE ALGORITHM                              *
;  255          0547  2 !                                                                     *
;  256          0548  2 ! GIVENS:  n = length of the divisor                                  *
;  257          0549  2 !          m = length of dividend - n                                 *
;  258          0550  2 !          radix = 10 (decimal)                                       *
;  259          0551  2 !                                                                     *
;  260          0552  2 ! STEP 1.  Normalize.  Set D = FLOOR (radix/(v1+1)) where v1 is  the  *
;  261          0553  2 !          first digit of the divisor which must not be zero.  Where  *
;  262          0554  2 !          U0 U1...Um+n represent the chunks of  15  digits  of  the  *
;  263          0555  2 !          dividend and V1 V2...Vn represent the chunks of 15 digits  *
;  264          0556  2 !          of the divisor.                                            *
;  265          0557  2 !          Multiply  A  by  D  thus  giving the sequence of 15 digit  *
;  266          0558  2 !          chunks U0 U1 U2...Um+n. (Note the introduction of the new  *
;  267          0559  2 !          chunk.)  Multiply B by d to obtain a sequence  of  chunks  *
;  268          0560  2 !          V1 V2...Vn. (Note no new chunk occurs                      *
;  269          0561  2 !                                                                     *
;  270          0562  2 ! STEP 2.  Set J = 0.  This is the value we will loop on.  For this   *
;  271          0563  2 !          routine we will loop "LOOP" number of times.  Steps  2-7   *
;  272          0564  2 !          will provide the basis for the division of Uj Uj+1...Uj+n  *
;  273          0565  2 !          by V1 V2...Vn, to get a single quotient digit - Qj.        *
;  274          0566  2 !                                                                     *
;  275          0567  2 ! STEP 3.  Calculate the first digit of the quotient:                *
;  276          0568  2 !          If Uj = V1 then set q = radix-1.  Otherwise, set q =       *
;  277          0569  2 !          FLOOR(Uj*radix + Uj+1)/V1).  Now test if V2*q >            *
;  278          0570  2 !          (((Uj*radix + Uj+1 - q*V1)*radix)+Uj+2).  If so, then      *
;  279          0571  2 !          decrease q by 1 and repeat this test.  When finish q is    *
;  280          0572  2 !          either equal to the qoutient digit or one greater.         *
;  281          0573  2 !                                                                     *
;  282          0574  2 ! STEP 4.  Multiply and subtract.  Replace Uj Uj+1...Uj+n by         *
;  283          0575  2 !          Uj Uj+1...Uj+n - (q * V1 V2...Vn).                         *
;  284          0576  2 !          This step consists of a simple multiplication by a one-place *
;  285          0577  2 !          number, combined with a subtraction.  The digits           *
;  286          0578  2 !          Uj Uj+1...Uj+n should be kept positive; if the result of this*
;  287          0579  2 !          step is negative, Uj Uj+1...Uj+n should be left as the true  *
;  288          0580  2 !          value plus radix raised to the n+1, i.e. as the radix'     *
;  289          0581  2 !          complement of the true value, and a "borrow" to the left   *
;  290          0582  2 !          should be remembered.                                      *
;  291          0583  2 !                                                                     *
;  292          0584  2 ! STEP 5.  Set Q[.J] = q.  This is a digit of the quotient.  If the   *
;  293          0585  2 !          result of STEP 4 was negative, go to STEP 6; otherwise go to *
;  294          0586  2 !          STEP 7.                                                    *
;  295          0587  2 !                                                                     *
;  296          0588  2 ! STEP 6.  Decrease Q[.J] by 1. Add 0V1 V2...Vn to Uj Uj+1...Uj+n.    *
;  297          0589  2 !                                                                     *
;  298          0590  2 ! STEP 7.  Loop on J.  If J <= "LOOP" then go back to STEP 3.         *
;  299          0591  2 !                                                                     *
;  300          0592  2 !**********************************************************************
```

```
;  302          0593  2 !+
;  303          0594  2 ! First check the decimal overflow setting of the flags word found in
;  304          0595  2 ! the BASIC frame.  Use the setting found in the flags word to feed into
;  305          0596  2 ! the PSW (that value overrides the current setting found in the PSW).
;  306          0597  2 !-
;  307          0598  2
;  308          0599  2         FMP = .FP;                                      ! Get frame pointer
;  309          0600  2         SAVE_FP = .FMP[BSF$A_SAVED_FP];                 ! Get saved frame pointer
;  310          0601  2         IF .SAVE_FP[BSF$A_HANDLER] EQL BAS$HANDLER
;  311          0602  2           THEN
;  312          0603  3             BEGIN
;  313          0604  4             IF (((.SAVE_FP[BSF$W_FCD_FLAGS]) AND (BSF$M_FCD_DV)) NEQ 0)
;  314          0605  4               THEN
;  315          0606  4                 BISPSW (%REF(PSW$M_DV))               ! Set DV bit in PSW
;  316          0607  4               ELSE
;  317          0608  4                 BICPSW (%REF(PSW$M_DV));              ! Clear DV bit in PSW
;  318          0609  4             END
;  319          0610  3           ELSE
;  320          0611  3             BISPSW (%REF(PSW$M_DV));                  ! Set DV bit in PSW
;  321          0612  2 !+
;  322          0613  2 ! Unpack descripters and convert scaled decimal strings to character strings
;  323          0614  2 !-
;  324          0615  2         B_ADDR = QBUF + 95;
;  325          0616  2         SAV_BADDR = .B_ADDR ;
;  326          0617  2         LIB$$UNPACK_SD_R8 (.B, B_LENGTH, B_EXP, B_SIGN, B_ADDR);
;  327          0618  2         IF .B_LENGTH EQL 0
;  328          0619  2           THEN
;  329          0620  2             BAS$$STOP(BAS$K_DIVBY_ZER);
;  330          0621  2         A_ADDR = .SAV_BADDR - 33;
;  331          0622  2         LIB$$UNPACK_SD_R8 (.A, A_LENGTH, A_EXP, A_SIGN, A_ADDR);
;  332          0623  2 !+
;  333          0624  2 ! Calculate the resultant sign.
;  334          0625  2 !-
;  335          0626  2         C_SIGN = .A_SIGN XOR .B_SIGN;
;  336          0627  2 !+
;  337          0628  2 ! Calculate maximum number of result digits required
;  338          0629  2 !+
;  339          0630  2         Q_LENGTH = (.A_LENGTH + .A_EXP) - (.B_LENGTH + .B_EXP)
;  340          0631  2                  - .C[DSC$B_SCALE] + .RND_TRUNC/5;
;  341          0632  2         IF .Q_LENGTH LSS 0
;  342          0633  2           THEN
;  343          0634  2 !+
;  344          0635  2 ! Special case for zero quotient
;  345          0636  2 !-
;  346          0637  3             BEGIN
;  347          0638  3             MOVP (C[DSC$W_LENGTH], ZERO + 15 - .C[DSC$W_LENGTH]/2,
;  348          0639  3                   .C[DSC$A_POINTER]);
;  349          0640  3             END
;  350          0641  2           ELSE
;  351          0642  3             BEGIN
;  352          0643  3 !+
;  353          0644  3 ! Determine the number of digits required in A to obtain the proper number
;  354          0645  3 ! of digits in the result
;  355          0646  3 !-
;  356          0647  3             A_LEN = .B_LENGTH + .Q_LENGTH;
;  357          0648  3 !+
;  358          0649  3 ! Determine the number of 15 digit CHUNKS needed to hold B, the required
```

```
;  359          0650  3 ! digits of A and the result digits
;  360          0651  3 !-
;  361          0652  3             A_CHUNKS = (.A_LEN + 14)/15;
;  362          0653  3             B_CHUNKS = (.B_LENGTH + 14)/15;
;  363          0654  3             Q_CHUNKS = (.Q_LENGTH + 29)/15;
;  364          0655  3 !+
;  365          0656  3 ! For the algorithm we must have A_CHUNKS >= B_CHUNKS + Q_CHUNKS.
;  366          0657  3 !-
;  367          0658  3             A_CHUNKS = MAXU(.A_CHUNKS, .B_CHUNKS + .Q_CHUNKS);
;  368          0659  3 !+
;  369          0660  3 ! Convert A and B strings to packed decimal arrays.
;  370          0661  3 !-
;  371          0662  3             LIB$$CVT_STR_PACK_R9 (.A_ADDR, .A_LENGTH, .A_CHUNKS, ABUF8);
;  372          0663  3             MOVP (%REF(15), ZERO, ABUF);
;  373          0664  3             LIB$$CVT_STR_PACK_R9 (.B_ADDR, .B_LENGTH, .B_CHUNKS, BBUF);
```

```
;  375     0665  3 !+
;  376     0666  3 ! Step 1 - Normalize A and B.  NOTE: If B_CHUNKS = 1 this step is not necessary
;  377     0667  3 ! and the computation of q can be simplified.  A flag is used to indicate the
;  378     0668  3 ! proper method of evaluating q.  FLAG = 1 if .B_CHUNKS = 1 and 0 otherwise.
;  379     0669  3 !-
;  380     0670  3             IF .B_CHUNKS NEQ 1
;  381     0671  3               THEN
;  382     0672  4                 BEGIN
;  383     0673  4                 FLAG = 0;
;  384     0674  4                 STATUS = LIB$$CALC_D_R7 (BBUF, QBUF);
;  385     0675  4                 IF .STATUS NEQ 1                    ! STATUS = 1 <==> D = 1
;  386     0676  4                   THEN
;  387     0677  5                     BEGIN
;  388     0678  5                     LIB$$MUL_PACK_R10 (QBUF, ABUF8, .A_CHUNKS, .A_CHUNKS+1,
;  389     0679  5                                        ABUF8);
;  390     0680  5                     LIB$$MUL_PACK_R10 (QBUF, BBUF, .B_CHUNKS, .B_CHUNKS, BBUF);
;  391     0681  4                     END;
;  392     0682  4                 END
;  393     0683  3               ELSE
;  394     0684  3                 FLAG = 1;
;  395     0685  3
;  396     0686  3 !+
;  397     0687  3 ! Ready to start the actual divide algorithm.
;  398     0688  3 !-
;  399     0689  3             INCR J FROM 0 TO (.Q_CHUNKS*8 - 8) BY 8 DO
;  400     0690  4               BEGIN
;  401     0691  4 !+
;  402     0692  4 ! Step 3 - Calculate digit of quotient.
;  403     0693  4 !-
;  404     0694  4                 LIB$$CALC_Q_R9 (BBUF, ABUF + .J, .FLAG, QBUF + .J);
;  405     0695  4 !+
;  406     0696  4 ! Step 4 - Multiply and subtract.  Replace the digits of ABUF by ABUF - Q*BBUF
;  407     0697  4 !-
;  408     0698  4                 LIB$$MUL_PACK_R10 (QBUF+.J, BBUF, .B_CHUNKS, .B_CHUNKS+1, QBBUF8);
;  409     0699  4                 STATUS = LIB$$SUB_PACK_R8 (.B_CHUNKS, ABUF + .J, QBBUF);
;  410     0700  4 !+
;  411     0701  4 ! Step 6 - Adjust q if the result of step 4 was negative
;  412     0702  4 !-
;  413     0703  4                 IF .STATUS EQL 1                    ! If remainder is negative
;  414     0704  4                   THEN
;  415     0705  4                     LIB$$ADJUST_Q_R9 (.B_CHUNKS, ABUF8 + .J, BBUF, QBUF + .J);
;  416     0706  3               END;
;  417     0707  3
;  418     0708  3 !+
;  419     0709  3 ! Convert the array of 15 digit chunks of the quotient to one packed decimal
;  420     0710  3 ! string of the length specified by the C descripter and copy the result to the
;  421     0711  3 ! location pointed to by the C descripter.
;  422     0712  3 !-
;  423     0713  3           STATUS = LIB$$CVT_AP_P_R8 (.RND_TRUNC,
;  424     0714  3                                      .C,
;  425     0715  3                                      QBUF,
;  426     0716  3                                      .Q_CHUNKS,
;  427     0717  3                                     (.Q_LENGTH - (.RND_TRUNC/5)) - 15*(.Q_CHUNKS-1),
;  428     0718  3                                      .C_SIGN);
;  429     0719  3           IF .STATUS EQL 1
;  430     0720  3             THEN
;  431     0721  3               BAS$$STOP(BAS$K_DECERR)
```

```
;  432          0722  2          END;
;  433          0723  1 END;


                                                        .TITLE  BAS$EXTEND_DIVP
                                                        .IDENT  \1-005\

                                                        .PSECT  _BAS$CODE,NOWRT,  SHR,  PIC,2

                                        00# 00000 P.AAA:    .BYTE   0[15]                                                   ;
                                        0C  0000F           .BYTE   12                                                      ;

                                                ZERO=               P.AAA
                                                        .EXTRN  BAS$$STOP, LIB$$CALC_D_R7
                                                        .EXTRN  LIB$$CALC_Q_R9, LIB$$SOB_PACK_R8
                                                        .EXTRN  LIB$$CVT_AP_P_R8
                                                        .EXTRN  LIB$$UNPACK_SD_R8
                                                        .EXTRN  LIB$$MUL_PACK_R10
                                                        .EXTRN  LIB$$ADJUST_Q_R9
                                                        .EXTRN  LIB$$CVT_STR_PACK_R9
                                                        .EXTRN  BAS$HANDLER, BAS$K_DECERR
                                                        .EXTRN  BAS$K_DIVBY_ZER

                                        0FFC 00000          .ENTRY  BAS$EXTEND_DIVP, Save R2,R3,R4,R5,R6,R7,R8,-            ; 0453
                                                                    R9,R10,R11                                              ;
                                    5E      FF34    CE  9E 00002        MOVAB   -204(SP), SP                                ;
                                14  AE              5D  D0 00007        MOVL    FP, FMP                                     ; 0599
                                50      14      AE  D0 0000B        MOVL    FMP, R0                                     ; 0600
                                10  AE      0C     A0  D0 0000F        MOVL    12(R0), SAVE_FP                             ;
                                50      10      AE  D0 00014        MOVL    SAVE_FP, R0                                 ; 0601
                                51  00000000G  00  9E 00018        MOVAB   BAS$HANDLER, R1                             ;
                                51              60  D1 0001F        CMPL    (R0), R1                                    ;
                                                0B  12 00022        BNEQ    1$                                          ;
                            06      E6      A0      0A  E0 00024        BBS     #10, -26(R0), 1$                            ; 0604
                                        0080    8F  B9 00029        BICPSW  #128                                        ; 0608
                                                04  11 0002D        BRB     2$                                          ; 0601
                                        0080    8F  B8 0002F 1$:    BISPSW  #128                                        ; 0611
                                    EC  AD          7F  AE  9E 00033 2$:    MOVAB   QBUF+95, B_ADDR                     ; 0615
                                    FC  AD          EC  AD  D0 00038        MOVL    B_ADDR, SAV_BADDR                   ; 0616
                                        58          EC  AD  9E 0003D        MOVAB   B_ADDR, R8                          ; 0617
                                        57          EA  AD  9E 00041        MOVAB   B_SIGN, R7                          ;
                                        56          EB  AD  9E 00045        MOVAB   B_EXP, R6                           ;
                                        55          F0  AD  9E 00049        MOVAB   B_LENGTH, R5                        ;
                                        54          08  AC  D0 0004D        MOVL    B, R4                               ;
                                            00000000G   00  16 00051        JSB     LIB$$UNPACK_SD_R8                   ;
                                        5A          F0  AD  3C 00057        MOVZWL  B_LENGTH, R10                       ; 0618
                                                    0D  12 0005B        BNEQ    3$                                  ;
                                            00000000G   8F  DD 0005D        PUSHL   #BAS$K_DIVBY_ZER                    ; 0620
                                00000000G   00          01  FB 00063        CALLS   #1, BAS$$STOP                       ;
                            F4  AD          FC  AD      21  C3 0006A 3$:    SUBL3   #33, SAV_BADDR, A_ADDR              ; 0621
                                        58          F4  AD  9E 00070        MOVAB   A_ADDR, R8                          ; 0622
                                        57          F2  AD  9E 00074        MOVAB   A_SIGN, R7                          ;
                                        56          F3  AD  9E 00078        MOVAB   A_EXP, R6                           ;
                                        55          FA  AD  9E 0007C        MOVAB   A_LENGTH, R5                        ;
                                        54          04  AC  D0 00080        MOVL    A, R4                               ;
                                            00000000G   00  16 00084        JSB     LIB$$UNPACK_SD_R8                   ;
                            E9  AD          F2  AD      EA  AD  8D 0008A        XORB3   B_SIGN, A_SIGN, C_SIGN              ; 0626
```

```
                                51        FA    AD   3C 00091          MOVZWL  A_LENGTH, R1                                    ; 0630
                                50        F3    AD   98 00095          CVTBL   A_EXP, R0                                       ;
                                51              50   C0 00099          ADDL2   R0, R1                                          ;
                                50        EB    AD   98 0009C          CVTBL   B_EXP, R0                                       ;
                                50              5A   C0 000A0          ADDL2   RTO, R0                                         ;
                                51              50   C2 000A3          SUBL2   R0, R1                                          ;
                     52         0C        AC         08 C1 000A6          ADDL3   #8, C, R2                                       ; 0631
                                50              62   98 000AB          CVTBL   (R2), R0                                        ;
                                51              50   C2 000AE          SUBL2   R0, R1                                          ;
                0C   AE         10        AC         05 C7 000B1          DIVL3   #5, RND_TRUNC, 12(SP)                           ;
                                D8        AD      0C BE41 9E 000B7          MOVAB   @12(SP)[R1], Q_LENGTH                           ;
                                                     1A 18 000BD          BGEQ    4$                                              ; 0632
                                50      FF3C      CF   9E 000BF          MOVAB   ZERO+15, R0                                     ; 0638
                                54        0C      BC   3C 000C4          MOVZWL  @C, R4                                          ;
                                54              02   C6 000C8          DIVL2   #2, R4                                          ;
                                50              54   C2 000CB          SUBL2   R4, R0                                          ;
                     55         0C        AC         04 C1 000CE          ADDL3   #4, C, R5                                       ; 0639
                     95                   60      0C BC   34 000D3          MOVP    @C, (R0), @(R5)+                                ;
                                                     04 000D8          RET                                                     ; 0632
                                E4        AD      D8 BD4A 9E 000D9 4$:      MOVAB   @Q_LENGTH[R10], A_LEN                           ; 0647
                     50         E4        AD         0E C1 000DF          ADDL3   #14, A_LEN, R0                                  ; 0652
                E0   AD                             0F C7 000E4          DIVL3   #15, R0, A_CHUNKS                               ;
                                57        0E      AA   9E 000E9          MOVAB   14(R10), R7                                     ; 0653
                DC   AD                             0F C7 000ED          DIVL3   #15, R7, B_CHUNKS                               ;
                     57         D8        AD         1D C1 000F2          ADDL3   #29, Q_LENGTH, R7                               ; 0654
                D4   AD                             0F C7 000F7          DIVL3   #15, R7, Q_CHUNKS                               ;
                     58         DC        AD      D4 AD C1 000FC          ADDL3   Q_CHUNKS, B_CHUNKS, R8                          ; 0658
                                50        E0    AD   D0 00102          MOVL    A_CHUNKS, R0                                    ;
                                58              50   D1 00106          CMPL    R0, R8                                          ;
                                                03   1E 00109          BGEQU   5$                                              ;
                                50              58   D0 0010B          MOVL    R8, R0                                          ;
                                   E0     AD         50   D0 0010E 5$:      MOVL    R0, A_CHUNKS                                    ;
                                59              30   AE   9E 00112          MOVAB   ABUF8, R9                                       ; 0662
                                58              E0   AD   D0 00116          MOVL    A_CHUNKS, R8                                    ;
                                57              FA   AD   3C 0011A          MOVZWL  A_LENGTH, R7                                    ;
                                56              F4   AD   D0 0011E          MOVL    A_ADDR, R6                                      ;
                                      00000000G       00   16 00122          JSB     LIB$$CVT_STR_PACK_R9                            ;
                28   AE         FEC3      CF         0F   34 00128          MOVP    #15, ZERO, ABUF                                 ; 0663
                                59              68   AE   9E 0012F          MOVAB   BBUF, R9                                        ; 0664
                                58              DC   AD   D0 00133          MOVL    B_CHUNKS, R8                                    ;
                                57              5A   D0   00137          MOVL    RTO, R7                                         ;
                                56              EC   AD   D0 0013A          MOVL    B_ADDR, R6                                      ;
                                      00000000G       00   16 0013E          JSB     LIB$$CVT_STR_PACK_R9                            ;
                                01              DC   AD   D1 00144          CMPL    B_CHUNKS, #1                                    ; 0670
                                                     52   13 00148          BEQL    6$                                              ;
                                              18   AE   D4   0014A          CLRL    FLAG                                            ; 0673
                                57            20   AE   9E   0014D          MOVAB   QBUF, R7                                        ; 0674
                                56            68   AE   9E   00151          MOVAB   BBUF, R6                                        ;
                                      00000000G       00   16 00155          JSB     LIB$$CALC_D_R7                                  ;
                           1C   AE                   50   D0 0015B          MOVL    R0, STATUS                                      ;
                                01            1C   AE   D1   0015F          CMPL    STATUS, #1                                      ; 0675
                                                     3B   13 00163          BEQL    7$                                              ;
                                5A            30   AE   9E   00165          MOVAB   ABUF8, R10                                      ; 0678
                     59         E0        AD         01   C1 00169          ADDL3   #1, A_CHUNKS, R9                                ;
                                57            30   AE   9E   0016E          MOVAB   ABUF8, R7                                       ;
                                56            20   AE   9E   00172          MOVAB   QBUF, R6                                        ;
                                58            E0   AD   D0   00176          MOVL    A_CHUNKS, R8                                    ;
```

```
                                00000000G   00   16 0017A            JSB     LIB$$MUL_PACK_R10                       ;
                              5A        68   AE   9E 00180            MOVAB   BBUF, R10                               ; 0680
                              57        68   AE   9E 00184            MOVAB   BBUF, R7                                ;
                              56        20   AE   9E 00188            MOVAB   QBUF, R6                                ;
                              59        DC   AD   D0 0018C            MOVL    B_CHUNKS, R9                            ;
                              58        DC   AD   D0 00190            MOVL    B_CHUNKS, R8                            ;
                                00000000G   00   16 00194            JSB     LIB$$MUL_PACK_R10                       ;
                                            04   11 0019A            BRB     7$                                      ; 0670
                           18   AE           01   D0 0019C 6$:       MOVL    #1, FLAG                                ; 0684
                08   AE     D4   AD           03   78 001A0 7$:       ASHL    #3, Q_CHUNKS, 8(SP)                     ; 0689
                           08   AE           08   C2 001A6            SUBL2   #8, 8(SP)                               ;
                04   AE     DC   AD           01   C1 001AA            ADDL3   #1, B_CHUNKS, 4(SP)                     ; 0698
                              5B            08   CE 001B0            MNEGL   #8, J                                   ;
                                            67   11 001B3            BRB     9$                                      ;
                              6E        20 AE4B   9E 001B5 8$:       MOVAB   QBUF[J], (SP)                           ; 0694
                              57        28 AE4B   9E 001BA            MOVAB   ABUF[J], R7                             ;
                              56        68   AE   9E 001BF            MOVAB   BBUF, R6                                ;
                              59        6E   D0 001C3                MOVL    (SP), R9                                ;
                              58        18   AE   D0 001C6            MOVL    FLAG, R8                                ;
                                00000000G   00   16 001CA            JSB     LIB$$CALC_Q_R9                          ;
                              5A        BC   AD   9E 001D0            MOVAB   QBBUF8, R10                             ; 0698
                              57        68   AE   9E 001D4            MOVAB   BBUF, R7                                ;
                              59        04   AE   D0 001D8            MOVL    4(SP), R9                               ;
                              58        DC   AD   D0 001DC            MOVL    B_CHUNKS, R8                            ;
                              56        6E   D0 001E0                MOVL    (SP), R6                                ;
                                00000000G   00   16 001E3            JSB     LIB$$MUL_PACK_R10                       ;
                              58        B4   AD   9E 001E9            MOVAB   QBBUF, R8                               ; 0699
                              57        28 AE4B   9E 001ED            MOVAB   ABUF[J], R7                             ;
                              56        DC   AD   D0 001F2            MOVL    B_CHUNKS, R6                            ;
                                00000000G   00   16 001F6            JSB     LIB$$SUB_PACK_R8                        ;
                           1C   AE           50   D0 001FC            MOVL    R0, STATUS                              ;
                              01        1C   AE   D1 00200            CMPL    STATUS, #1                              ; 0703
                                            16   12 00204            BNEQ    9$                                      ;
                              58        68   AE   9E 00206            MOVAB   BBUF, R8                                ; 0705
                              57        30 AE4B   9E 0020A            MOVAB   ABUF8[J], R7                            ;
                              59        6E   D0 0020F                MOVL    (SP), R9                                ;
                              56        DC   AD   D0 00212            MOVL    B_CHUNKS, R6                            ;
                                00000000G   00   16 00216            JSB     LIB$$ADJUST_Q_R9                        ;
                FF92          5B            08   08   AE   F1 0021C 9$:       ACBL    8(SP), #8, J, 8$                ; 0689
                              57        D8   AD   0C   AE   C3 00223            SUBL3   12(SP), Q_LENGTH, R7            ; 0717
                              56        D4   AD           0F   C5 00229            MULL3   #15, Q_CHUNKS, R6               ;
                                            57           56   C2 0022E            SUBL2   R6, R7                          ;
                                            57           0F   C0 00231            ADDL2   #15, R7                         ;
                                            55        20   AE   9E 00234            MOVAB   QBUF, R5                        ; 0713
                                            58        E9   AD   9A 00238            MOVZBL  C_SIGN, R8                      ;
                                            56        D4   AD   D0 0023C            MOVL    Q_CHUNKS, R6                    ;
                                            54        0C   AC   D0 00240            MOVL    C, R4                           ;
                                            53        10   AC   D0 00244            MOVL    RND_TRUNC, R3           ;
                                00000000G   00   16 00248            JSB     LIB$$CVT_AP_P_R8                        ;
                           1C   AE           50   D0 0024E            MOVL    R0, STATUS                              ;
                              01        1C   AE   D1 00252            CMPL    STATUS, #1                              ; 0719
                                            0D   12 00256            BNEQ    10$                                     ;
                                00000000G   8F   DD 00258            PUSHL   #BAS$K_DECERR                           ; 0721
                      00000000G   00            01   FB 0025E            CALLS   #1, BAS$$STOP                           ;
                                                04 00265 10$:       RET                                             ; 0723

; Routine Size:  614 bytes,    Routine Base:  _BAS$CODE + 0010
```

```
;  434          0724  1 END                                   ! end of module BAS$EXTEND_DIVP
;  435          0725  1
;  436          0726  0 ELUDOM



;                                PSECT SUMMARY
;
;      Name                    Bytes                           Attributes
;
;  _BAS$CODE                     630  NOVEC,NOWRT,  RD ,  EXE,  SHR,  LCL,  REL,  CON,  PIC,ALIGN(2)


;                              Library Statistics
;
;                                        --------  Symbols  --------      Pages       Processing
;      File                              Total     Loaded   Percent       Mapped      Time
;
;  _$255$DUA28:[SYSLIB]STARLET.L32;1      9776          4         0          581        00:01.0



;                                       COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LIS$:BASEDIVP/OBJ=OBJ$:BASEDIVP MSRC$:BASEDIVP/UPDATE=(ENH$:BASEDIVP)

; Size:          614 code + 16 data bytes
; Run Time:          00:11.7
; Elapsed Time:      00:26.4
; Lines/CPU Min:      3713
; Lexemes/CPU-Min: 13815
; Memory Used:  161 pages
; Compilation Complete
```

BASENDDFS
LIS

BASERROR
LIS

BASENDDEF
LIS

BASEDIT
LIS

BASEND
LIS

BASEDIVP
LIS

BASEMULP
LIS

BASERRTXT
LIS

BASENDGSB
LIS