```
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    SSSSSSSS    CCCCCCCC    AAAAAA    LL          EEEEEEEEEE
BBBBBBBB      AAAAAA      SSSSSSSS    SSSSSSSS    CCCCCCCC    AAAAAA    LL          EEEEEEEEEE
BB      BB   AA    AA   SS          SS          CC          AA    AA  LL          EE
BB      BB   AA    AA   SS          SS          CC          AA    AA  LL          EE
BB      BB   AA    AA   SS          SS          CC          AA    AA  LL          EE
BB      BB   AA    AA   SS          SS          CC          AA    AA  LL          EE
BBBBBBBB     AA    AA     SSSSSS      SSSSSS    CC          AA    AA  LL          EEEEEEEE
BBBBBBBB     AA    AA     SSSSSS      SSSSSS    CC          AA    AA  LL          EEEEEEEE
BB      BB   AAAAAAAAAA         SS          SS  CC          AAAAAAAAAA  LL          EE
BB      BB   AAAAAAAAAA         SS          SS  CC          AAAAAAAAAA  LL          EE
BB      BB   AA    AA           SS          SS  CC          AA    AA  LL          EE              ....
BB      BB   AA    AA           SS          SS  CC          AA    AA  LL          EE              ....
BBBBBBBB     AA    AA   SSSSSSSS    SSSSSSSS      CCCCCCCC  AA    AA  LLLLLLLLLL  EEEEEEEEEE      ....
BBBBBBBB     AA    AA   SSSSSSSS    SSSSSSSS      CCCCCCCC  AA    AA  LLLLLLLLLL  EEEEEEEEEE      ....

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$SCALE (                          ! Support scaling
;     2          0002  0                 IDENT = '1-008'             ! File: BASSCALE.B32 Edit: PLL1008
;     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 !++
;    32          0032  1 ! FACILITY:  BASIC-PLUS-2 Miscellaneous
;    33          0033  1 !
;    34          0034  1 ! ABSTRACT:
;    35          0035  1 !
;    36          0036  1 !       This module contains compiled code support routines
;    37          0037  1 !       for scaling and descaling.
;    38          0038  1 !
;    39          0039  1 ! ENVIRONMENT:  VAX-11 user mode
;    40          0040  1 !
;    41          0041  1 ! AUTHOR: John Sauter, CREATION DATE: 08-MAY-1979
;    42          0042  1 !
;    43          0043  1 ! MODIFIED BY:
;    44          0044  1 !
;    45          0045  1 ! 1-001 - Original.
;    46          0046  1 ! 1-002 - Add BAS$$SCALE_R1.  JBS 29-MAY-1979
;    47          0047  1 ! 1-003 - Use BAS$$COPY_D to fetch double-precision numbers.
;    48          0048  1 !           JBS 29-MAY-1979
;    49          0049  1 ! 1-004 - Add BAS$$SCALE_L_R1.  JBS 26-JUN-1979
;    50          0050  1 ! 1-005 - Change MTH$FLOOR_D to MTH$DFLOOR.  JBS 27-JUL-1979
;    51          0051  1 ! 1-006 - Change BAS$$COPY_D to BAS$$COPY_D_R1.  JBS 23-AUG-1979
;    52          0052  1 ! 1-007 - Change MTH$DFLOOR to MTH$DINT.  JBS 19-DEC-1979
;    53          0053  1 ! 1-008 - Remove the check for a BASIC frame from BAS$SCALE_D_R1
;    54          0054  1 !         and BAS$DSCALE_D_R1.  (This is so BAS$CHANGE can call
;    55          0055  1 !         them.)  PLL 22-MAY-1981
;    56          0056  1 !--
;    57          0057  1
```

```
;   58          0058  1 !<BLF/PAGE>
```

```
;    60          0059  1 !
;    61          0060  1 ! SWITCHES:
;    62          0061  1 !
;    63          0062  1
;    64          0063  1 SWITCHES ADDRESSING_MODE (EXTERNAL = GENERAL, NONEXTERNAL = WORD_RELATIVE);
;    65          0064  1
;    66          0065  1 !
;    67          0066  1 ! LINKAGES:
;    68          0067  1 !
;    69          0068  1
;    70          0069  1 REQUIRE 'RTLIN:BASLNK';                         ! BASIC linkages
;    71          0146  1
;    72          0147  1 LINKAGE
;    73          0148  1     COPY_JSB = JSB (REGISTER = 0, REGISTER = 1) :       !
;    74          0149  1     NOTUSED (2, 3, 4, 5, 6, 7, 8, 9, 10, 11);
;    75          0150  1
;    76          0151  1 !
;    77          0152  1 ! TABLE OF CONTENTS:
;    78          0153  1 !
;    79          0154  1
;    80          0155  1 FORWARD ROUTINE
;    81          0156  1     BAS$SCALE_D_R1 : NOVALUE BAS$SCALE_LINK,    ! Scale a number
;    82          0157  1     BAS$DSCALE_D_R1 : NOVALUE BAS$SCALE_LINK,   ! Descale a number
;    83          0158  1     BAS$$SCALE_RT : NOVALUE BAS$SCALE_JSB,      ! Fetch the scale as double
;    84          0159  1     BAS$$SCALE_L_R1 : BAS$SCALE_JSB;            ! Fetch the scale power
;    85          0160  1
;    86          0161  1 !
;    87          0162  1 ! INCLUDE FILES:
;    88          0163  1 !
;    89          0164  1
;    90          0165  1 REQUIRE 'RTLIN:BASFRAME';                       ! BSF symbols
;    91          0368  1
;    92          0369  1 LIBRARY 'RTLSTARLE';                            ! symbols for strings
;    93          0370  1
;    94          0371  1 REQUIRE 'RTLIN:RTLPSECT';                       ! macros for defing psects
;    95          0466  1
;    96          0467  1 !
;    97          0468  1 ! MACROS:
;    98          0469  1 !
;    99          0470  1 !       NONE
;   100          0471  1 !
;   101          0472  1 ! EQUATED SYMBOLS:
;   102          0473  1 !
;   103          0474  1 !       NONE
;   104          0475  1 !
;   105          0476  1 ! PSECTS:
;   106          0477  1 !
;   107          0478  1 DECLARE_PSECTS (BAS);                           ! declare psects for BAS$ facility
;   108          0479  1 !
;   109          0480  1 ! OWN STORAGE:
;   110          0481  1 !
;   111          0482  1 !       NONE
;   112          0483  1 !
;   113          0484  1 ! EXTERNAL REFERENCES:
;   114          0485  1 !
;   115          0486  1
;   116          0487  1 EXTERNAL ROUTINE
```

```
;  117          0488  1     MTH$DINT,                          ! Remove fraction part
;  118          0489  1     BAS$$COPY_D_R1 : COPY_JSB NOVALUE,  ! Move a D_float number
;  119          0490  1     BAS$$MULD,                         ! Multiply D_float numbers
;  120          0491  1     BAS$$DIVD,                         ! Divide D_float numbers
;  121          0492  1     BAS$HANDLER;                       ! BASIC frame marker
;  122          0493  1
```

```
;  124          0494  1 GLOBAL ROUTINE BAS$SCALE_D_R1 (                       ! Scale a value
;  125          0495  1         VALHI,                                        ! High 32-bits of value
;  126          0496  1         VALLO                                         ! Low 32-bits of value
;  127          0497  1     ) : NOVALUE BAS$SCALE_LINK =
;  128          0498  1
;  129          0499  1 !++
;  130          0500  1 ! FUNCTIONAL DESCRIPTION:
;  131          0501  1 !
;  132          0502  1 !       Scale a value.  This is done by multiplying by the scale factor
;  133          0503  1 !       and integerizing the result.
;  134          0504  1 !
;  135          0505  1 !
;  136          0506  1 ! FORMAL PARAMETERS:
;  137          0507  1 !
;  138          0508  1 !       VAL.rd.v        The D_floating value to be scaled, presented to
;  139          0509  1 !                       BLISS as VALHI and VALLO.
;  140          0510  1 !
;  141          0511  1 ! IMPLICIT INPUTS:
;  142          0512  1 !
;  143          0513  1 !       The scale factor, in the major frame.
;  144          0514  1 !
;  145          0515  1 ! IMPLICIT OUTPUTS:
;  146          0516  1 !
;  147          0517  1 !       NONE
;  148          0518  1 !
;  149          0519  1 ! ROUTINE VALUE:
;  150          0520  1 !
;  151          0521  1 !       The scaled, double precision number.
;  152          0522  1 !
;  153          0523  1 ! COMPLETION CODES:
;  154          0524  1 !
;  155          0525  1 !       NONE
;  156          0526  1 !
;  157          0527  1 ! SIDE EFFECTS:
;  158          0528  1 !
;  159          0529  1 !       May get arithmetic faults.
;  160          0530  1 !
;  161          0531  1 !--
;  162          0532  1
;  163          0533  2     BEGIN
;  164          0534  2
;  165          0535  2     EXTERNAL REGISTER
;  166          0536  2         BSF$A_MAJOR_STG : REF BLOCK [0, BYTE] FIELD (BSF$MAJOR_FRAME),
;  167          0537  2         BSF$A_MINOR_STG,
;  168          0538  2         BSF$A_TEMP_STG;
;  169          0539  2
;  170          0540  2
;  171          0541  2     LOCAL
;  172          0542  2         VAL : VECTOR [2, LONG];                     ! D_floating value being manipulated
;  173          0543  2
;  174          0544  2 !+
;  175          0545  2 ! Save the input value, so we get some registers to work with.
;  176          0546  2 !-
;  177          0547  2     VAL [0] = .VALHI;
;  178          0548  2     VAL [1] = .VALLO;
;  179          0549  2
;  180          0550  3         BEGIN
```

```
;  181          0551  3 !+
;  182          0552  3 ! Multiply the argument by the scale factor, and then integerize.
;  183          0553  3 !-
;  184          0554  3
;  185          0555  3         LOCAL
;  186          0556  3             MAJOR_FMP : REF BLOCK [, BYTE] FIELD (BSF$FCD);
;  187          0557  3
;  188          0558  3         MAJOR_FMP = BSF$A_MAJOR_STG [BSF$FRAME_BASE];
;  189          0559  3         BAS$$MULD (VAL [0], MAJOR_FMP [BSF$D_SCALE_DOU], VAL [0]);
;  190          0560  3         MTH$DINT (VAL [0]);
;  191          0561  2         END;
;  192          0562  2     RETURN;
;  193          0563  1     END;                                    ! of BAS$SCALE_D_R1


                                                        .TITLE  BAS$SCALE
                                                        .IDENT  \1-008\

                                                        .EXTRN  MTH$DINT, BAS$$COPY_D_R1
                                                        .EXTRN  BAS$$MULD, BAS$$DIVD
                                                        .EXTRN  BAS$HANDLER

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

                        5E              04  C2 00000 BAS$SCALE_D_R1::
                                                    SUBL2   #4, SP                          ; 0494
                                        50  DD 00003        PUSHL   VALHI                   ; 0547
                                    04  AE  51  D0 00005    MOVL    VALLO, VAL+4            ; 0548
                                        50      00C3  CB 9E 00009    MOVAB   195(R11), MAJOR_FMP     ; 0558
                                                      5E DD 0000E    PUSHL   SP              ; 0559
                                                   D0 A0 9F 00010    PUSHAB  -48(MAJOR_FMP)  ;
                                                   08 AE 9F 00013    PUSHAB  VAL             ;
                                00000000G       00      03 FB 00016    CALLS   #3, BAS$$MULD   ;
                                                        5E DD 0001D    PUSHL   SP              ; 0560
                                00000000G       00      01 FB 0001F    CALLS   #1, MTH$DINT    ;
                                                        5E  08 C0 00026    ADDL2   #8, SP          ; 0563
                                                            05 00029    RSB                     ;

; Routine Size:  42 bytes,    Routine Base:  _BAS$CODE + 0000

;  194          0564  1
```

```
;  196          0565  1 GLOBAL ROUTINE BAS$DSCALE_D_R1 (                      ! Descale a value
;  197          0566  1         VALHI,                                        ! High 32-bits of value
;  198          0567  1         VALLO                                         ! Low 32-bits of value
;  199          0568  1     ) : NOVALUE BAS$SCALE_LINK =
;  200          0569  1
;  201          0570  1 !++
;  202          0571  1 ! FUNCTIONAL DESCRIPTION:
;  203          0572  1 !
;  204          0573  1 !       Descale a value.  This is done by dividing by the scale factor.
;  205          0574  1 !
;  206          0575  1 !
;  207          0576  1 !
;  208          0577  1 ! FORMAL PARAMETERS:
;  209          0578  1 !
;  210          0579  1 !       VAL.rd.v        The D_floating value to be descaled, presented
;  211          0580  1 !                       to BLISS as VALHI and VALLO.
;  212          0581  1 !
;  213          0582  1 ! IMPLICIT INPUTS:
;  214          0583  1 !
;  215          0584  1 !       The scale factor, in the major frame.
;  216          0585  1 !
;  217          0586  1 ! IMPLICIT OUTPUTS:
;  218          0587  1 !
;  219          0588  1 !       NONE
;  220          0589  1 !
;  221          0590  1 ! ROUTINE VALUE:
;  222          0591  1 !
;  223          0592  1 !       The descaled, double precision number.
;  224          0593  1 !
;  225          0594  1 ! COMPLETION CODES:
;  226          0595  1 !
;  227          0596  1 !       NONE
;  228          0597  1 !
;  229          0598  1 ! SIDE EFFECTS:
;  230          0599  1 !
;  231          0600  1 !       May get arithmetic faults.
;  232          0601  1 !
;  233          0602  1 !--
;  234          0603  1
;  235          0604  2     BEGIN
;  236          0605  2
;  237          0606  2     EXTERNAL REGISTER
;  238          0607  2         BSF$A_MAJOR_STG : REF BLOCK [0, BYTE] FIELD (BSF$MAJOR_FRAME),
;  239          0608  2         BSF$A_MINOR_STG,
;  240          0609  2         BSF$A_TEMP_STG;
;  241          0610  2
;  242          0611  2
;  243          0612  2     LOCAL
;  244          0613  2         VAL : VECTOR [2, LONG];                     ! D_floating value being manipulated
;  245          0614  2
;  246          0615  2 !+
;  247          0616  2 ! Save the input value, so we get some registers to work with.
;  248          0617  2 !-
;  249          0618  2     VAL [0] = .VALHI;
;  250          0619  2     VAL [1] = .VALLO;
;  251          0620  2
;  252          0621  3         BEGIN
```

```
;  253          0622  3 !+
;  254          0623  3 ! Divide the argument by the scale factor, then load into R0 and R1.
;  255          0624  3 !-
;  256          0625  3
;  257          0626  3         LOCAL
;  258          0627  3             MAJOR_FMP : REF BLOCK [, BYTE] FIELD (BSF$FCD);
;  259          0628  3
;  260          0629  3         MAJOR_FMP = BSF$A_MAJOR_STG [BSF$FRAME_BASE];
;  261          0630  3         BAS$$DIVD (MAJOR_FMP [BSF$D_SCALE_DOU], VAL [0], VAL [0]);
;  262          0631  4         BEGIN
;  263          0632  4
;  264          0633  4         REGISTER
;  265          0634  4             R0 = 0,
;  266          0635  4             R1 = 1;
;  267          0636  4
;  268          0637  4         R0 = .VAL [0];
;  269          0638  4         R1 = .VAL [1];
;  270          0639  3         END;
;  271          0640  2         END;
;  272          0641  2     RETURN;
;  273          0642  1     END;                                    ! of BAS$DSCALE_D_R1



                             5E              04  C2 00000 BAS$DSCALE_D_R1::
                                                           SUBL2   #4, SP                          ; 0565
                                             50  DD 00003         PUSHL   VALHI                   ; 0618
                                 04  AE      51  D0 00005         MOVL    VALLO, VAL+4            ; 0619
                                 50      00C3 CB  9E 00009         MOVAB   195(R11), MAJOR_FMP     ; 0629
                                             5E  DD 0000E         PUSHL   SP                      ; 0630
                                     04  AE  9F  00010         PUSHAB  VAL                     ;
                                     D0  A0  9F  00013         PUSHAB  -48(MAJOR_FMP)          ;
                          00000000G      00  03  FB 00016         CALLS   #3, BAS$$DIVD           ;
                                 50          8E  7D 0001D         MOVQ    VAL, R0                 ; 0637
                                             05  00020         RSB                             ; 0642

; Routine Size:  33 bytes,    Routine Base:  _BAS$CODE + 002A

;  274          0643  1
```

```
;  276     0644  1 GLOBAL ROUTINE BAS$$SCALE_R1 (                    ! Fetch the scale
;  277     0645  1         FMP                                       ! Frame containing scale
;  278     0646  1     ) : NOVALUE BAS$SCALE_JSB =
;  279     0647  1
;  280     0648  1 !++
;  281     0649  1 ! FUNCTIONAL DESCRIPTION:
;  282     0650  1 !
;  283     0651  1 !       Fetch the scale value from a frame.  This routine is for use by
;  284     0652  1 !       math routines to fetch the scale from their caller.  If the
;  285     0653  1 !       frame is not a BASIC frame, a double-precision 1.0 is returned.
;  286     0654  1 !
;  287     0655  1 ! FORMAL PARAMETERS:
;  288     0656  1 !
;  289     0657  1 !       FMP.ra.v        The (possibly BASIC) frame containing the scale
;  290     0658  1 !                       factor.
;  291     0659  1 !
;  292     0660  1 ! IMPLICIT INPUTS:
;  293     0661  1 !
;  294     0662  1 !       The scale factor, in the major frame.
;  295     0663  1 !
;  296     0664  1 ! IMPLICIT OUTPUTS:
;  297     0665  1 !
;  298     0666  1 !       NONE
;  299     0667  1 !
;  300     0668  1 ! ROUTINE VALUE:
;  301     0669  1 !
;  302     0670  1 !       The scale factor, as a double-precision number.
;  303     0671  1 !
;  304     0672  1 ! COMPLETION CODES:
;  305     0673  1 !
;  306     0674  1 !       NONE
;  307     0675  1 !
;  308     0676  1 ! SIDE EFFECTS:
;  309     0677  1 !
;  310     0678  1 !       May get arithmetic faults.
;  311     0679  1 !
;  312     0680  1 !--
;  313     0681  1
;  314     0682  2     BEGIN
;  315     0683  2
;  316     0684  2     MAP
;  317     0685  2         FMP : REF BLOCK [, BYTE] FIELD (BSF$FCD);
;  318     0686  2
;  319     0687  2 !+
;  320     0688  2 ! If this is not a BASIC frame, return 1.0.
;  321     0689  2 !-
;  322     0690  2
;  323     0691  3     IF (.FMP [BSF$A_HANDLER] NEQA BAS$HANDLER)
;  324     0692  2     THEN
;  325     0693  3         BEGIN
;  326     0694  3
;  327     0695  3         REGISTER
;  328     0696  3             R0 = 0,
;  329     0697  3             R1 = 1;
;  330     0698  3
;  331     0699  3         BUILTIN
;  332     0700  3             CVTLD;
```

```
;  333          0701  3
;  334          0702  3          CVTLD (%REF (1), R0);
;  335          0703  3          END
;  336          0704  2      ELSE
;  337          0705  3          BEGIN
;  338          0706  3      !+
;  339          0707  3      ! Otherwise return the real scale factor.
;  340          0708  3      !-
;  341          0709  3
;  342          0710  3          LOCAL
;  343          0711  3              MAJOR_FMP : REF BLOCK [, BYTE] FIELD (BSF$FCD),
;  344          0712  3              BSF$A_MAJOR_STG : REF BLOCK [, BYTE] FIELD (BSF$MAJOR_FRAME),
;  345          0713  3              VAL : VECTOR [2, LONG];
;  346          0714  3
;  347          0715  3          BSF$A_MAJOR_STG = .FMP [BSF$A_BASE_R11];
;  348          0716  3          MAJOR_FMP = BSF$A_MAJOR_STG [BSF$FRAME_BASE];
;  349          0717  3          BAS$$COPY_D_R1 (MAJOR_FMP [BSF$D_SCALE_DOU], VAL);
;  350          0718  4          BEGIN
;  351          0719  4
;  352          0720  4          REGISTER
;  353          0721  4              R0 = 0,
;  354          0722  4              R1 = 1;
;  355          0723  4
;  356          0724  4          R0 = .VAL [0];
;  357          0725  4          R1 = .VAL [1];
;  358          0726  3          END;
;  359          0727  2          END;
;  360          0728  2
;  361          0729  2      RETURN;
;  362          0730  1      END;                                    ! of BAS$SCALE_R1


                                                     5E            08  C2 00000 BAS$$SCALE_R1::
                                                                               SUBL2   #8, SP                          ; 0644
                                                     51            50  D0 00003         MOVL    R0, R1
                                                     50 00000000G  00  9E 00006         MOVAB   BAS$HANDLER, R0         ; 0691
                                                     50            61  D1 0000D         CMPL    (FMP), R0
                                                                   05  13 00010         BEQL    1$
                                                     50            01  6E 00012         CVTLD   #1, R0                  ; 0702
                                                                   15  11 00015         BRB     2$                  ; 0691
                                                     50       F4   A1  D0 00017 1$:     MOVL    -12(FMP), BSF$A_MAJOR_STG ; 0715
                                                     51            6E  9E 0001B         MOVAB   VAL, R1                 ; 0717
                                                     50      0093  C0  9E 0001E         MOVAB   147(R0), R0
                                                        00000000G  00  16 00023         JSB     BAS$$COPY_D_R1
                                                     50            6E  7D 00029         MOVQ    VAL, R0                 ; 0724
                                                     5E            08  C0 0002C 2$:     ADDL2   #8, SP                  ; 0730
                                                                   05  0002F         RSB

; Routine Size:  48 bytes,    Routine Base:  _BAS$CODE + 004B

;  363          0731  1
```

```
;  365     0732  1 GLOBAL ROUTINE BAS$$SCALE_L_R1 (                    ! Fetch the scale
;  366     0733  1         FMP                                         ! Frame containing scale
;  367     0734  1     ) : BAS$SCALE_JSB =
;  368     0735  1
;  369     0736  1 !++
;  370     0737  1 ! FUNCTIONAL DESCRIPTION:
;  371     0738  1 !
;  372     0739  1 !       Fetch the scale value from a frame.  This routine is for use by
;  373     0740  1 !       math routines to fetch the scale from their caller.  If the
;  374     0741  1 !       frame is not a BASIC frame, 0 is returned.
;  375     0742  1 !
;  376     0743  1 ! FORMAL PARAMETERS:
;  377     0744  1 !
;  378     0745  1 !       FMP.ra.v        The (possibly BASIC) frame containing the scale
;  379     0746  1 !                       factor.
;  380     0747  1 !
;  381     0748  1 ! IMPLICIT INPUTS:
;  382     0749  1 !
;  383     0750  1 !       The scale factor, in the major frame.
;  384     0751  1 !
;  385     0752  1 ! IMPLICIT OUTPUTS:
;  386     0753  1 !
;  387     0754  1 !       NONE
;  388     0755  1 !
;  389     0756  1 ! ROUTINE VALUE:
;  390     0757  1 !
;  391     0758  1 !       The scale factor, as an integer power of 10.
;  392     0759  1 !
;  393     0760  1 ! COMPLETION CODES:
;  394     0761  1 !
;  395     0762  1 !       NONE
;  396     0763  1 !
;  397     0764  1 ! SIDE EFFECTS:
;  398     0765  1 !
;  399     0766  1 !       NONE
;  400     0767  1 !
;  401     0768  1 !--
;  402     0769  1
;  403     0770  2     BEGIN
;  404     0771  2
;  405     0772  2     MAP
;  406     0773  2         FMP : REF BLOCK [, BYTE] FIELD (BSF$FCD);
;  407     0774  2
;  408     0775  2 !+
;  409     0776  2 ! If this is not a BASIC frame, return 0.
;  410     0777  2 !-
;  411     0778  2
;  412     0779  3     IF (.FMP [BSF$A_HANDLER] NEQA BAS$HANDLER)
;  413     0780  2     THEN
;  414     0781  2         0
;  415     0782  2     ELSE
;  416     0783  3         BEGIN
;  417     0784  3 !+
;  418     0785  3 ! Otherwise return the real scale factor.
;  419     0786  3 !-
;  420     0787  3
;  421     0788  3         LOCAL
```

```
;  422          0789  3            MAJOR_FMP : REF BLOCK [, BYTE] FIELD (BSF$FCD),
;  423          0790  3            BSF$A_MAJOR_STG : REF BLOCK [, BYTE] FIELD (BSF$MAJOR_FRAME),
;  424          0791  3            VAL : VECTOR [2, LONG];
;  425          0792  3
;  426          0793  3        BSF$A_MAJOR_STG = .FMP [BSF$A_BASE_R11];
;  427          0794  3        MAJOR_FMP = BSF$A_MAJOR_STG [BSF$FRAME_BASE];
;  428          0795  3        .MAJOR_FMP [BSF$B_SCA_V_DOU]
;  429          0796  3        END
;  430          0797  3
;  431          0798  1    END;                                    ! of BAS$SCALE_D_R1


                                5E             08  C2 00000 BAS$$SCALE_L_R1::
                                                           SUBL2   #8, SP                          ; 0732
                                51 00000000G   00  9E 00003         MOVAB   BAS$HANDLER, R1         ; 0779
                                51             60  D1 0000A         CMPL    (FMP), R1               ;
                                               04  13 0000D         BEQL    1$                      ;
                                               50  D4 0000F         CLRL    R0                      ;
                                               0D  11 00011         BRB     2$                      ;
                                50       F4    A0  DO 00013 1$:     MOVL    -12(FMP), BSF$A_MAJOR_STG       ; 0793
                                50     00C3    C0  9E 00017         MOVAB   195(R0), MAJOR_FMP      ; 0794
                                50       CD    A0  98 0001C         CVTBL   -51(MAJOR_FMP), R0      ; 0795
                                5E             08  C0 00020 2$:     ADDL2   #8, SP                  ; 0798
                                                   05 00023         RSB                             ;

; Routine Size:  36 bytes,    Routine Base:  _BAS$CODE + 007B

;  432          0799  1
;  433          0800  1 END
;  434          0801  1
;  435          0802  0 ELUDOM
```

PSECT SUMMARY

Name	Bytes	Attributes
_BAS$CODE	159	NOVEC,NOWRT, RD , EXE, SHR, LCL, REL, CON, PIC,ALIGN(2)

Library Statistics

File	Symbols Total	Symbols Loaded	Symbols Percent	Pages Mapped	Processing Time
_255DUA28:[SYSLIB]STARLET.L32;1	9776	0	0	581	00:01.0

```
;                                       COMMAND QUALIFIERS

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

; Size:         159 code + 0 data bytes
; Run Time:       00:07.9
; Elapsed Time:   00:21.2
; Lines/CPU Min:   6075
; Lexemes/CPU-Min: 21098
; Memory Used:  57 pages
; Compilation Complete
```

BASRTDIM LIS
BASSARITH LIS
BASSCALE LIS
BASSIGNAL LIS
BASRUNINI LIS
BASSCRATC LIS
BASRSTSFI LIS
BASSLEEP LIS
BASSTOP LIS
BASSEG LIS