```
PPPPPPPPPPPP        AAAAAAAAA        SSSSSSSSSSS  RRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
PPPPPPPPPPPP        AAAAAAAAA        SSSSSSSSSSS  RRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
PPPPPPPPPPPP        AAAAAAAAA        SSSSSSSSSSS  RRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
PPP          PPP   AAA         AAA  SSS          RRR         RRR        TTT        LLL
PPP          PPP   AAA         AAA  SSS          RRR         RRR        TTT        LLL
PPP          PPP   AAA         AAA  SSS          RRR         RRR        TTT        LLL
PPP          PPP   AAA         AAA  SSS          RRR         RRR        TTT        LLL
PPP          PPP   AAA         AAA  SSS          RRR         RRR        TTT        LLL
PPP          PPP   AAA         AAA  SSS          RRR         RRR        TTT        LLL
PPPPPPPPPPPP       AAA         AAA     SSSSSSSSS     RRRRRRRRRRR        TTT        LLL
PPPPPPPPPPPP       AAA         AAA     SSSSSSSSS     RRRRRRRRRRR        TTT        LLL
PPPPPPPPPPPP       AAA         AAA     SSSSSSSSS     RRRRRRRRRRR        TTT        LLL
PPP                AAAAAAAAAAAAAAA           SSS   RRR   RRR          TTT        LLL
PPP                AAAAAAAAAAAAAAA           SSS   RRR   RRR          TTT        LLL
PPP                AAAAAAAAAAAAAAA           SSS   RRR   RRR          TTT        LLL
PPP                AAA         AAA           SSS   RRR      RRR       TTT        LLL
PPP                AAA         AAA           SSS   RRR      RRR       TTT        LLL
PPP                AAA         AAA           SSS   RRR      RRR       TTT        LLL
PPP                AAA         AAA  SSSSSSSSSSS   RRR         RRR     TTT        LLLLLLLLLLLLLLL
PPP                AAA         AAA  SSSSSSSSSSS   RRR         RRR     TTT        LLLLLLLLLLLLLLL
PPP                AAA         AAA  SSSSSSSSSSS   RRR         RRR     TTT        LLLLLLLLLLLLLLL
PPP                AAA         AAA  SSSSSSSSSSS   RRR         RRR     TTT        LLLLLLLLLLLLLLL
```

```
PPPPPPPP      AAAAAA      SSSSSSSS  WW      WW  RRRRRRRR      IIIIII    FFFFFFFFFF  DDDDDDDD        11
PPPPPPPP      AAAAAA      SSSSSSSS  WW      WW  RRRRRRRR      IIIIII    FFFFFFFFFF  DDDDDDDD        11
PP      PP  AA      AA  SS        WW      WW  RR      RR      II      FF          DD      DD    1111
PP      PP  AA      AA  SS        WW      WW  RR      RR      II      FF          DD      DD    1111
PP      PP  AA      AA  SS        WW      WW  RR      RR      II      FF          DD      DD      11
PP      PP  AA      AA  SS        WW      WW  RR      RR      II      FF          DD      DD      11
PPPPPPPP    AA      AA    SSSSSS  WW      WW  RRRRRRRR      II      FFFFFFF     DD      DD      11
PPPPPPPP    AA      AA    SSSSSS  WW      WW  RRRRRRRR      II      FFFFFFF     DD      DD      11
PP          AAAAAAAAAA          SS  WW  WW  WW  RR  RR      II      FF          DD      DD      11
PP          AAAAAAAAAA          SS  WW  WW  WW  RR  RR      II      FF          DD      DD      11
PP          AA      AA          SS  WWWW  WWWW  RR    RR    II      FF          DD      DD      11        ....
PP          AA      AA          SS  WWWW  WWWW  RR    RR    II      FF          DD      DD      11        ....
PP          AA      AA  SSSSSSSS    WW      WW  RR      RR  IIIIII  FF          DDDDDDDD      111111      ....
PP          AA      AA  SSSSSSSS    WW      WW  RR      RR  IIIIII  FF          DDDDDDDD      111111      ....

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 PAS$WRITE_REALF_D1 ( %TITLE 'Write an D_floating in F format - V1 semantics'
;     2          0002  0                       IDENT = '1-002'                 ! File: PASWRIFD1.B32 Edit: SBL1002
;     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 OTHEPWISE 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:     Pascal Language Support
;    32          0032  1 !
;    33          0033  1 ! ABSTRACT:
;    34          0034  1 !
;    35          0035  1 !       This module contains a procedure which writes a D_floating in
;    36          0036  1 !       fixed-point notation to a textfile using V1 semantics.
;    37          0037  1 !
;    38          0038  1 ! ENVIRONMENT:  User mode - AST reentrant
;    39          0039  1 !
;    40          0040  1 ! AUTHOR: Steven B. Lionel, CREATION DATE: 1-April-1981
;    41          0041  1 !
;    42          0042  1 ! MODIFIED BY:
;    43          0043  1 !
;    44          0044  1 ! 1-001 - Original.  SBL 1-April-1981
;    45          0045  1 ! 1-002 - Make total-width a longword.  SBL 29-June-1982
;    46          0046  1 !--
;    47          0047  1
```

```
;    49          0048  1 %SBTTL 'Declarations'
;    50          0049  1 !
;    51          0050  1 ! PROLOGUE DEFINITIONS:
;    52          0051  1 !
;    53          0052  1
;    54          0053  1 REQUIRE 'RTLIN:PASPROLOG';                    ! Externals, linkages, PSECTs, structures
;    55          0117  1
;    56          0118  1 !
;    57          0119  1 ! TABLE OF CONTENTS:
;    58          0120  1 !
;    59          0121  1
;    60          0122  1 FORWARD ROUTINE
;    61          0123  1     PASSWRITE_REALF_D1: NOVALUE,               ! Write to textfile
;    62          0124  1     LOCAL_HANDLER;                             ! Local handler
;    63          0125  1
;    64          0126  1 !
;    65          0127  1 ! MACROS:
;    66          0128  1 !
;    67          0129  1 !       NONE
;    68          0130  1 !
;    69          0131  1 ! EQUATED SYMBOLS:
;    70          0132  1 !
;    71          0133  1 !       NONE
;    72          0134  1 !
;    73          0135  1 ! FIELDS:
;    74          0136  1 !
;    75          0137  1 !       NONE
;    76          0138  1 !
;    77          0139  1 ! OWN STORAGE:
;    78          0140  1 !
;    79          0141  1 !       NONE
;    80          0142  1 !
```

```
;    82          0143  1 %SBTTL 'PAS$WRITE_REALF_D1 - Write D_floating in F format to textfile - V1'
;    83          0144  1 GLOBAL ROUTINE PAS$WRITE_REALF_D1 (
;    84          0145  1         PFV: REF $PAS$PFV_FILE_VARIABLE,                ! File variable
;    85          0146  1         VALUE_0: BLOCK [4, BYTE],VALUE_1,               ! Value to write
;    86          0147  1         TOTAL_WIDTH: SIGNED,                            ! Total field width
;    87          0148  1         FRAC_DIGITS: SIGNED,                            ! Digits in fraction
;    88          0149  1         ERROR                                           ! Error unwind address
;    89          0150  1     ): NOVALUE =
;    90          0151  1
;    91          0152  1 !++
;    92          0153  1 ! FUNCTIONAL DESCRIPTION:
;    93          0154  1 !
;    94          0155  1 !       This procedure writes a D_floating value in fixed-point notation
;    95          0156  1 !       to the specified textfile.
;    96          0157  1 !
;    97          0158  1 !       It uses the VAX-11 Pascal V1 semantics where if the value is not
;    98          0159  1 !       negative, an extra leading blank appears.  This is contary to
;    99          0160  1 !       the ISO standard.
;   100          0161  1 !
;   101          0162  1 !       This procedure is implemented using public "single-dollar" interfaces
;   102          0163  1 !       to the Pascal Run-Time Library so that it may be excluded from
;   103          0164  1 !       the shareable image PASRTL.EXE.
;   104          0165  1 !
;   105          0166  1 ! CALLING SEQUENCE:
;   106          0167  1 !
;   107          0168  1 !       CALL PAS$WRITE_REALF_D1 (PFV.mr.r, VALUE.rd.v , TOTAL_WIDTH.rl.v,
;   108          0169  1 !                                FRAC_DIGITS.rl.v [, ERROR.ja.r]])
;   109          0170  1 !
;   110          0171  1 ! FORMAL PARAMETERS:
;   111          0172  1 !
;   112          0173  1 !       PFV             - The Pascal File Variable (PFV) passed by reference.
;   113          0174  1 !                         The structure of the PFV is defined in PASPFV.REQ.
;   114          0175  1 !
;   115          0176  1 !       VALUE           - The D_floating value to write.  Note that this is
;   116          0177  1 !                         passed by immediate value, requiring two argument
;   117          0178  1 !                         list positions.
;   118          0179  1 !
;   119          0180  1 !       TOTAL_WIDTH     - Total field width.
;   120          0181  1 !
;   121          0182  1 !       FRAC_DIGITS     - Number of digits in fraction.
;   122          0183  1 !
;   123          0184  1 !       ERROR           - Optional.  Address to unwind to if an error occurs.
;   124          0185  1 !
;   125          0186  1 ! IMPLICIT INPUTS:
;   126          0187  1 !
;   127          0188  1 !       NONE
;   128          0189  1 !
;   129          0190  1 ! IMPLICIT OUTPUTS:
;   130          0191  1 !
;   131          0192  1 !       NONE
;   132          0193  1 !
;   133          0194  1 ! ROUTINE VALUE:
;   134          0195  1 !
;   135          0196  1 !       NONE
;   136          0197  1 !
;   137          0198  1 ! SIDE EFFECTS:
;   138          0199  1 !
```

```
;  139          0200  1 !       If the file is the standard file OUTPUT, it is implicitly opened.
;  140          0201  1 !
;  141          0202  1 ! SIGNALLED ERRORS:
;  142          0203  1 !
;  143          0204  1 !       LINTOOLON - line too long
;  144          0205  1 !       NEGWIDDIG - negative Width or Digits specification is not allowed
;  145          0206  1 !
;  146          0207  1 !--
;  147          0208  1
;  148          0209  2     BEGIN
;  149          0210  2
;  150          0211  2     BUILTIN
;  151          0212  2         ACTUALCOUNT;
;  152          0213  2
;  153          0214  2     LOCAL
;  154          0215  2         FIELD_WIDTH,                                    ! Minimum/actual field width
;  155          0216  2         TEMP_STRING,                                    ! Temporary for convert
;  156          0217  2         PFV_ADDR: VOLATILE,                             ! Enable argument
;  157          0218  2         UNWIND_ACT: VOLATILE,                           ! Enable argument
;  158          0219  2         ERROR_ADDR: VOLATILE;                           ! Enable argument
;  159          0220  2
;  160          0221  2     ENABLE
;  161          0222  2         LOCAL_HANDLER (PFV_ADDR, UNWIND_ACT, ERROR_ADDR);       ! Enable error handler
;  162          0223  2
;  163          0224  2     !+
;  164          0225  2     ! Get ERROR parameter, if present.
;  165          0226  2     !-
;  166          0227  2
;  167          0228  2     IF ACTUALCOUNT () GEQU 6
;  168          0229  2     THEN
;  169          0230  2         ERROR_ADDR = .ERROR;                ! Set unwind address
;  170          0231  2
;  171          0232  2     PFV_ADDR = PFV [PFV$R_PFV];             ! Set PFV address
;  172          0233  2
;  173          0234  2     !+
;  174          0235  2     ! The difference between V2 and V1 semantics is that for V2
;  175          0236  2     ! (and ISO), the representation of the floating point value
;  176          0237  2     ! has no leading blank if not negative.  V1 put a leading
;  177          0238  2     ! blank there, possibly causing the field to be extended
;  178          0239  2     ! an extra character.
;  179          0240  2     !
;  180          0241  2     ! To implement this, do a trial conversion of the value into
;  181          0242  2     ! a scratch string.  The convert routine will tell us how many
;  182          0243  2     ! characters are required for the conversion.  If the value
;  183          0244  2     ! is not negative, increase the field width to 1 more character
;  184          0245  2     ! than is necessary.  This will cause an extra leading blank.
;  185          0246  2     !-
;  186          0247  2
;  187          0248  2     FIELD_WIDTH = .TOTAL_WIDTH; ! Get specified field width
;  188          0249  2
;  189          0250  2     IF (NOT .VALUE_0 [0,15,1,0]) AND    ! If value is not negative
;  190          0251  2        (.FIELD_WIDTH GEQ 0) AND         ! Don't bother for invalid width
;  191          0252  3        (.FRAC_DIGITS GEQ 0)             ! Don't bother for invalid digits
;  192          0253  2     THEN
;  193          0254  3         BEGIN
;  194          0255  3
;  195          0256  3         FIELD_WIDTH = 0;                ! Expand field
```

```
;  196          0257  3
;  197          0258  3          !+
;  198          0259  3          ! Do the conversion into a temporary.  FIELD_WIDTH will then have
;  199          0260  3          ! the number of characters needed.
;  200          0261  3          !-
;  201          0262  3
;  202          0263  3          PAS$CVT_D_T (VALUE_0,              ! Value to convert
;  203          0264  3                       TEMP_STRING,         ! Scratch string
;  204          0265  3                       FIELD_WIDTH,         ! Minimum/actual width
;  205          0266  3                       0,                   ! Maximum width
;  206          0267  3                       .FRAC_DIGITS);       ! Number of fraction digits
;  207          0268  3
;  208          0269  3          !+
;  209          0270  3          ! Set FIELD_WIDTH to be the maximum of the needed length plus 1
;  210          0271  3          ! and the caller-specified width.
;  211          0272  3          !-
;  212          0273  3
;  213          0274  3          FIELD_WIDTH = .FIELD_WIDTH + 1;
;  214          0275  3          IF .FIELD_WIDTH LSS .TOTAL_WIDTH
;  215          0276  3          THEN
;  216          0277  3              FIELD_WIDTH = .TOTAL_WIDTH;
;  217          0278  3
;  218          0279  2          END;
;  219          0280  2
;  220          0281  2      !+
;  221          0282  2      ! Now call PAS$WRITE_REALF_D to do the actual write.
;  222          0283  2      !-
;  223          0284  2
;  224          0285  2      PAS$WRITE_REALF_D (PFV [PFV$R_PFV], .VALUE_0, .VALUE_1,
;  225          0286  2                         .FIELD_WIDTH, .FRAC_DIGITS);
;  226          0287  2
;  227          0288  2      RETURN;
;  228          0289  2
;  229          0290  1      END;                                  ! End of routine PAS$WRITE_REALF_D1


                                .TITLE  PAS$WRITE_REALF_D1 Write an D_floating in F for
;                                                          mat - V1 semantics
                                .IDENT  \1-002\

                                .EXTRN  PAS$WRITE_REALF_D1
                                .EXTRN  PAS$CVT_D_T, PAS$WRITE_REALF_D

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

                          0000 00000            .ENTRY  PAS$WRITE_REALF_D1, Save nothing                ; 0144
                       5E      14   C2 00002    SUBL2   #20, SP                                         ;
                          08   AE   7C 00005    CLRQ    ERROR_ADDR                                      ; 0209
                          10   AE   D4 00008    CLRL    PFV_ADDR                                        ;
                       6D 005A CF   DE 0000B    MOVAL   3$, (FP)                                        ;
                       06          6C   91 00010    CMPB    (AP), #6                                    ; 0228
                                   05   1F 00013    BLSSU   1$                                          ;
                    08 AE      18   AC   D0 00015    MOVL    ERROR, ERROR_ADDR                           ; 0230
                    10 AE      04   AC   D0 0001A 1$: MOVL    PFV, PFV_ADDR                               ; 0232
                       6E      10   AC   D0 0001F    MOVL    TOTAL_WIDTH, FIELD_WIDTH                    ; 0248
                               09   AC   95 00023    TSTB    VALUE_0+1                                   ; 0250
                                    2C   19 00026    BLSS    2$                                          ;
```

```
                                        6E  D5 00028          TSTL    FIELD_WIDTH                             ; 0251
                                        28  19 0002A          BLSS    2$
                                    14  AC  D5 0002C          TSTL    FRAC_DIGITS                             ; 0252
                                        23  19 0002F          BLSS    2$
                                        6E  D4 00031          CLRL    FIELD_WIDTH                             ; 0256
                                    14  AC  DD 00033          PUSHL   FRAC_DIGITS                             ; 0267
                                        7E  D4 00036          CLRL    -(SP)                                   ; 0263
                                    08  AE  9F 00038          PUSHAB  FIELD_WIDTH
                                    10  AE  9F 0003B          PUSHAB  TEMP_STRING
                                    08  AC  9F 0003E          PUSHAB  VALUE_0
                          00000000G 00      05  FB 00041      CALLS   #5, PAS$CVT_D_T
                                        6E  D6 00048          INCL    FIELD_WIDTH                             ; 0274
                               10   AC      6E  D1 0004A      CMPL    FIELD_WIDTH, TOTAL_WIDTH                ; 0275
                                        04  18 0004E          BGEQ    2$
                                    6E      10  AC  D0 00050  MOVL    TOTAL_WIDTH, FIELD_WIDTH                ; 0277
                                    14  AC  DD 00054 2$:      PUSHL   FRAC_DIGITS                             ; 0286
                                    04  AE  DD 00057          PUSHL   FIELD_WIDTH
                                    7E      08  AC  7D 0005A  MOVQ    VALUE_0, -(SP)                          ; 0285
                                    04  AC  DD 0005E          PUSHL   PFV
                          00000000G 00      05  FB 00061      CALLS   #5, PAS$WRITE_REALF_D
                                        04  00068            RET                                             ; 0290
                                    0000 00069 3$:            .WORD   Save nothing                            ; 0209
                                    50      08  AC  D0 0006B  MOVL    8(AP), R0
                                    50      04  A0  D0 0006F  MOVL    4(R0), R0
                                    F4  A0  9F 00073          PUSHAB  ERROR_ADDR
                                    F8  A0  9F 00076          PUSHAB  UNWIND_ACT
                                    FC  A0  9F 00079          PUSHAB  PFV_ADDR
                                        03  DD 0007C          PUSHL   #3
                                        5E  DD 0007E          PUSHL   SP
                                    7E      04  AC  7D 00080  MOVQ    4(AP), -(SP)
                               0000V CF         03  FB 00084 CALLS   #3, LOCAL_HANDLER
                                        04  00089            RET

; Routine Size:  138 bytes,    Routine Base:  _PAS$CODE + 0000

;  230          0291  1
;  231          0292  1 !<BLF/PAGE>
```

```
;  233          0293  1 %SBTTL 'LOCAL_HANDLER - Local handler'
;  234          0294  1 ROUTINE LOCAL_HANDLER (
;  235          0295  1         SIGNAL_ARGS: REF BLOCK [, BYTE],        ! Signal arguments array
;  236          0296  1         MECH_ARGS: REF BLOCK [, BYTE],          ! Mechanism arguments array
;  237          0297  1         ENABLE_ARGS: REF VECTOR [, LONG]        ! Enable arguments array
;  238          0298  1     ) =
;  239          0299  1
;  240          0300  1 !++
;  241          0301  1 ! FUNCTIONAL DESCRIPTION:
;  242          0302  1 !
;  243          0303  1 !       This is the condition handler enabled by PAS$WRITE_REALF_D1.
;  244          0304  1 !       If the current signal is a Pascal error on the file our establisher
;  245          0305  1 !       was called with, we unwind to the caller of the establisher
;  246          0306  1 !       with R0 being the status code of the error.
;  247          0307  1 !
;  248          0308  1 !
;  249          0309  1 ! CALLING SEQUENCE:
;  250          0310  1 !
;  251          0311  1 !       status.wlc.v = STATUS_HANDLER (SIGNAL_ARGS.rl.ra, MECH_ARGS.rl.ra,
;  252          0312  1 !                                      ENABLE_ARGS.rl.ra)
;  253          0313  1 !
;  254          0314  1 ! FORMAL PARAMETERS:
;  255          0315  1 !
;  256          0316  1 !       SIGNAL_ARGS     - The signal argument list.
;  257          0317  1 !
;  258          0318  1 !       MECH_ARGS       - The mechanism argument list.
;  259          0319  1 !
;  260          0320  1 !       ENABLE_ARGS     - An array with the following
;  261          0321  1 !                         format:
;  262          0322  1 !
;  263          0323  1 !                               +----------------+
;  264          0324  1 !                               |   ENB_COUNT    |  <-- ENABLE_ARGS
;  265          0325  1 !                               +----------------+
;  266          0326  1 !                               |  ENB_PFV_ADDR  |
;  267          0327  1 !                               +----------------+
;  268          0328  1 !                               | ENB_UNWIND_ACT |
;  269          0329  1 !                               +----------------+
;  270          0330  1 !                               | ENB_ERROR_ADDR |
;  271          0331  1 !                               +----------------+
;  272          0332  1 !
;  273          0333  1 !                             ENB_COUNT is the count of following enable arguments.
;  274          0334  1 !                             The count is always at least 2.
;  275          0335  1 !
;  276          0336  1 !                             ENB_PFV_ADDR - If non-zero, the address of a longword
;  277          0337  1 !                             containing the PFV our establisher is operating on.
;  278          0338  1 !
;  279          0339  1 !                             ENB_UNWIND_ACT - Specifies the action
;  280          0340  1 !                             to take on an unwind.  The values are:
;  281          0341  1 !                              PAS$K_UNWIND_NOP - Do nothing
;  282          0342  1 !                              PAS$K_UNWIND_UNLOCK - Unlock PFV
;  283          0343  1 !
;  284          0344  1 !                             ENB_ERROR_ADDR - Ignored here.
;  285          0345  1 !
;  286          0346  1 ! IMPLICIT INPUTS:
;  287          0347  1 !
;  288          0348  1 !       The signaller's PFV placed as the first FAO argument in the primary
;  289          0349  1 !       signalled message.
```

```
;  290          0350  1 !
;  291          0351  1 ! IMPLICIT OUTPUTS:
;  292          0352  1 !
;  293          0353  1 !       NONE
;  294          0354  1 !
;  295          0355  1 ! ROUTINE VALUE:
;  296          0356  1 !
;  297          0357  1 !       SS$_RESIGNAL
;  298          0358  1 !
;  299          0359  1 ! SIDE EFFECTS:
;  300          0360  1 !
;  301          0361  1 !       May cause an unwind.
;  302          0362  1 !
;  303          0363  1 !--
;  304          0364  1
;  305          0365  2     BEGIN
;  306          0366  2
;  307          0367  2     LITERAL
;  308          0368  2         ENB_COUNT = 0,          ! Count of enable arguments
;  309          0369  2         ENB_PFV_ADDR = 1,       ! Address of address of PFV
;  310          0370  2         ENB_UNWIND_ACT= 2,      ! Address of unwind action
;  311          0371  2         ENB_ERROR_ADDR = 3;     ! Address of address of unwind PC
;  312          0372  2
;  313          0373  2     !+
;  314          0374  2     ! Determine if this is an unwind.  If so, resignal.
;  315          0375  2     ! Otherwise, see if we should cause an unwind.
;  316          0376  2     !-
;  317          0377  2
;  318          0378  2     IF .SIGNAL_ARGS [CHF$L_SIG_NAME] EQLU SS$_UNWIND
;  319          0379  2     THEN
;  320          0380  2         RETURN SS$_RESIGNAL;
;  321          0381  2
;  322          0382  2     IF ..ENABLE_ARGS [ENB_ERROR_ADDR] NEQ 0     ! Error "continue specified?
;  323          0383  2     THEN
;  324          0384  3         BEGIN
;  325          0385  3
;  326          0386  3         LOCAL
;  327          0387  3             COND_NAME: BLOCK [4, BYTE], ! Primary condition name
;  328          0388  3             COND_CODE;                  ! Sequence number of error
;  329          0389  3
;  330          0390  3         !+
;  331          0391  3         ! Get primary condition name.
;  332          0392  3         !-
;  333          0393  3
;  334          0394  3         COND_NAME = .SIGNAL_ARGS [CHF$L_SIG_NAME];
;  335          0395  3
;  336          0396  3         !+
;  337          0397  3         ! Is this a PAS$ error?  If not, resignal.
;  338          0398  3         !-
;  339          0399  3
;  340          0400  3         IF .COND_NAME [STS$V_FAC_NO] NEQU PAS$_FACILITY
;  341          0401  3         THEN
;  342          0402  3             RETURN SS$_RESIGNAL;
;  343          0403  3
;  344          0404  3         !+
;  345          0405  3         ! See if the error message is one which is "trapped"
;  346          0406  3         ! by ERROR:=CONTINUE.  This is done by comparing the
```

```
;  347          0407  3          ! message number against a select range.
;  348          0408  3          !-
;  349          0409  3
;  350          0410  3          COND_CODE = .COND_NAME [STS$V_CODE];     ! Get error number
;  351          0411  3          IF .COND_CODE GEQU PAS$$K_MSGCONTLO AND  ! Lowest number
;  352          0412  3             .COND_CODE LEQU PAS$$K_MSGCONTHI      ! Highest number
;  353          0413  3          THEN
;  354          0414  4              BEGIN
;  355          0415  4
;  356          0416  4              !+
;  357          0417  4              ! See if the PFVs match.  The signaller's PFV is the
;  358          0418  4              ! first FAO parameter in the primary message.                                    ;
;  359          0419  4              !-
;  360          0420  4
;  361          0421  4              IF .SIGNAL_ARGS [12,0,32,0] EQLA ..ENABLE_ARGS [ENB_PFV_ADDR]                   ;
;  362          0422  4              THEN                                                                            ;
;  363          0423  4                  !+                                                                          ;
;  364          0424  4                  ! We want to unwind to the PC specified in the enable argument
;  365          0425  4                  ! error address.
;  366          0426  4                  !-
;  367          0427  4
;  368          0428  5                  BEGIN
;  369          0429  6                  IF NOT $UNWIND (NEWPC=..ENABLE_ARGS [ENB_ERROR_ADDR])
;  370          0430  5                  THEN
;  371          0431  5                      SIGNAL_STOP (PAS$_BUGCHECK,1,BUG_UNWINDFAIL);
;  372          0432  4                  END;
;  373          0433  3              END;
;  374          0434  2          END;
;  375          0435  2
;  376          0436  2      RETURN SS$_RESIGNAL;                        ! Resignal error
;  377          0437  2
;  378          0438  1      END;                                        ! End of routine LOCAL_HANDLER


                                                .EXTRN  PAS$_FACILITY, PAS$$K_MSGCONTLO
                                                .EXTRN  PAS$$K_MSGCONTHI
                                                .EXTRN  SYS$UNWIND, PAS$_BUGCHECK

                                        0004 00000 LOCAL_HANDLER:
                                                .WORD   Save R2                                                 ; 0294
                                51      04      AC  D0 00002            MOVL    SIGNAL_ARGS, R1                 ; 0378
                    00000920    8F      04      A1  D1 00006            CMPL    4(R1), #2336                    ;
                                                52  13 0000E            BEQL    1$                              ;
                                50      0C      AC  D0 00010            MOVL    ENABLE_ARGS, R0                 ; 0382
                                        0C      B0  D5 00014            TSTL    @12(R0)                         ;
                                                49  13 00017            BEQL    1$                              ;
                                52      04      A1  D0 00019            MOVL    4(R1), COND_NAME                ; 0394
        00G             52              0C          10  ED 0001D        CMPZV   #16, #12, COND_NAME, S^PAS$_FACILITY ; 0400
                                                    3E  12 00022        BNEQ    1$                              ;
        52              52              0C          03  EF 00024        EXTZV   #3, #12, COND_NAME, COND_CODE       ; 0410
                    00000000G   8F                  52  D1 00029        CMPL    COND_CODE, #PAS$$K_MSGCONTLO    ; 0411
                                                    30  1F 00030        BLSSU   1$                              ;
                    00000000G   8F                  52  D1 00032        CMPL    COND_CODE, #PAS$$K_MSGCONTHI    ; 0412
                                                    27  1A 00039        BGTRU   1$                              ;
                          04    B0      0C      A1  D1 0003B            CMPL    12(R1), @4(R0)                  ; 0421
                                                20  12 00040            BNEQ    1$                              ;
```

```
                                        0C   B0  DD 00042          PUSHL   @12(R0)                                 ; 0429
                                             7E  D4 00045          CLRL    -(SP)                                   ;
                          00000000G  00      02  FB 00047          CALLS   #2, SYS$UNWIND                          ;
                                     11      50  E8 0004E          BLBS    R0, 1$                                  ;
                                             03  DD 00051          PUSHL   #3                                      ; 0431
                                             01  DD 00053          PUSHL   #1                                      ;
                                 00000000G   8F  DD 00055          PUSHL   #PAS$_BUGCHECK                          ;
                          00000000G  00      03  FB 00058          CALLS   #3, LIB$STOP                            ;
                                     50  0918  8F  3C 00062 1$:     MOVZWL  #2328, R0                               ; 0436
                                             04  00067             RET                                             ; 0438

; Routine Size:  104 bytes,    Routine Base:  _PAS$CODE + 008A

;  379          0439  1
;  380          0440  1 !<BLF/PAGE>
```

```
;  382          0441  1 END                                        ! End of module PAS$WRITE_REALF_D1
;  383          0442  1
;  384          0443  0 ELUDOM


                                                                    .EXTRN  LIB$STOP

;                                  PSECT SUMMARY
;
;      Name                        Bytes                           Attributes
;
;  _PAS$CODE                          242  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          7         0          581        00:01.0
;  _$255$DUA28:[PASRTL.OBJ]PASLIB.L32;1             427         27         6           33        00:00.4


;                                        COMMAND QUALIFIERS

;       BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/NOTRACE/LIS=LIS$:PASWRIFD1/OBJ=OBJ$:PASWRIFD1 MSRC$:PASWRIFD1/UPDATE=(ENH$:PASWRIFD1
;       )

; Size:          242 code + 0 data bytes
; Run Time:        00:06.9
; Elapsed Time:    00:31.7
; Lines/CPU Min:    3852
; Lexemes/CPU-Min:  9495
; Memory Used:  61 pages
; Compilation Complete
```

PASREWRIT LIS

PASSIGNAL LIS

PASSOR LIS

PASTRUNCA LIS

PASUNLOCK LIS

PASWRIBOO LIS

PASVECTOR LIS

PASWRIBIN LIS

PASUNDEFI LIS

PASVALIDA LIS

PASWRIHEX LIS

PASVM LIS

PASWRIFFI LIS

PASTIME LIS

PASWRIFDI LIS

PASUPDATE LIS

PASWRIENU LIS

PASSTATUS LIS

PASUFB LIS

PASWRICHA LIS