```
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    SSSSSSSS  TTTTTTTTTT    AAAAAA    TTTTTTTTTT  UU      UU    SSSSSSSS
PPPPPPPP      AAAAAA      SSSSSSSS    SSSSSSSS  TTTTTTTTTT    AAAAAA    TTTTTTTTTT  UU      UU    SSSSSSSS
PP      PP   AA      AA  SS          SS            TT       AA      AA      TT       UU      UU  SS
PP      PP   AA      AA  SS          SS            TT       AA      AA      TT       UU      UU  SS
PP      PP   AA      AA  SS          SS            TT       AA      AA      TT       UU      UU  SS
PP      PP   AA      AA  SS          SS            TT       AA      AA      TT       UU      UU  SS
PPPPPPPP     AA      AA    SSSSSS      SSSSSS      TT       AA      AA      TT       UU      UU    SSSSSS
PPPPPPPP     AA      AA    SSSSSS      SSSSSS      TT       AA      AA      TT       UU      UU    SSSSSS
PP           AAAAAAAAAA          SS          SS    TT       AAAAAAAAAA      TT       UU      UU          SS
PP           AAAAAAAAAA          SS          SS    TT       AAAAAAAAAA      TT       UU      UU          SS
PP           AA      AA          SS          SS    TT       AA      AA      TT       UU      UU          SS      ....
PP           AA      AA          SS          SS    TT       AA      AA      TT       UU      UU          SS      ....
PP           AA      AA  SSSSSSSS    SSSSSSSS      TT       AA      AA      TT       UUUUUUUUUU  SSSSSSSS        ....
PP           AA      AA  SSSSSSSS    SSSSSSSS      TT       AA      AA      TT       UUUUUUUUUU  SSSSSSSS        ....

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$STATUS ( %TITLE 'Get status of last I/O operation'
;     2          0002  0                 IDENT = '1-001'                 ! File: PASSTATUS.B32 Edit: SBL1001
;     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:     Pascal Language Support
;    32          0032  1 !
;    33          0033  1 ! ABSTRACT:
;    34          0034  1 !
;    35          0035  1 !       This module contains a procedure which implements the
;    36          0036  1 !       VAX-11 Pascal STATUS function.
;    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 !--
;    46          0046  1
```

```
;    48          0047  1 %SBTTL 'Declarations'
;    49          0048  1 !
;    50          0049  1 ! PROLOGUE DEFINITIONS:
;    51          0050  1 !
;    52          0051  1
;    53          0052  1 REQUIRE 'RTLIN:PASPROLOG';                    ! Externals, linkages, PSECTs, structures
;    54          0116  1
;    55          0117  1 !
;    56          0118  1 ! TABLE OF CONTENTS:
;    57          0119  1 !
;    58          0120  1
;    59          0121  1 FORWARD ROUTINE
;    60          0122  1     PAS$STATUS,                               ! Return file status
;    61          0123  1     STATUS_HANDLER;                           ! Local handler
;    62          0124  1
;    63          0125  1 !
;    64          0126  1 ! MACROS:
;    65          0127  1 !
;    66          0128  1 !       NONE
;    67          0129  1 !
;    68          0130  1 ! EQUATED SYMBOLS:
;    69          0131  1 !
;    70          0132  1 !       NONE
;    71          0133  1 !
;    72          0134  1 ! FIELDS:
;    73          0135  1 !
;    74          0136  1 !       NONE
;    75          0137  1 !
;    76          0138  1 ! OWN STORAGE:
;    77          0139  1 !
;    78          0140  1 !       NONE
```

```
;    80          0141  1 %SBTTL 'PAS$STATUS - Return file status'
;    81          0142  1 GLOBAL ROUTINE PAS$STATUS (
;    82          0143  1         PFV: REF $PAS$PFV_FILE_VARIABLE         . File variable
;    83          0144  1     ) =
;    84          0145  1
;    85          0146  1 !++
;    86          0147  1 ! FUNCTIONAL DESCRIPTION:
;    87          0148  1 !
;    88          0149  1 !       This procedure implements the VAX-11 Pascal STATUS function.
;    89          0150  1 !       It returns an integer value which indicates the status of the
;    90          0151  1 !       last operation on the specified file.  A negative result indicates
;    91          0152  1 !       that the file is at EOF, but that no error has occurred.  A zero
;    92          0153  1 !       result indicates that no error has occurredm and the file is not
;    93          0154  1 !       at EOF.  A positive result indicates that the last operation
;    94          0155  1 !       on the file caused an error, and the specific integer value
;    95          0156  1 !       denotes which error occurred.
;    96          0157  1 !
;    97          0158  1 !       Symbols of the form PAS$K_xxx are defined that describe the
;    98          0159  1 !       result of PAS$STATUS.  They are:
;    99          0160  1 !               PAS$K_EOF       = -1
;   100          0161  1 !               PAS$K_SUCCESS   = 0
;   101          0162  1 !               PAS$K_abcmnoxyz = positive integer, where "abcmnoxyz"
;   102          0163  1 !                                 is the same as for the equivalent
;   103          0164  1 !                                 error condition code name.  For
;   104          0165  1 !                                 example, if the last error to occur
;   105          0166  1 !                                 on the file was PAS$_FAIGETLOC,
;   106          0167  1 !                                 PAS$STATUS will return PAS$K_FAIGETLOC.
;   107          0168  1 !
;   108          0169  1 ! CALLING SEQUENCE:
;   109          0170  1 !
;   110          0171  1 !       Status.wl.v = PAS$STATUS (PFV.mr.r)
;   111          0172  1 !
;   112          0173  1 ! FORMAL PARAMETERS:
;   113          0174  1 !
;   114          0175  1 !       PFV             - The Pascal File Variable (PFV) passed by reference.
;   115          0176  1 !                         The structure of the PFV is defined in PASPFV.REQ.
;   116          0177  1 !
;   117          0178  1 ! IMPLICIT INPUTS:
;   118          0179  1 !
;   119          0180  1 !       NONE
;   120          0181  1 !
;   121          0182  1 ! IMPLICIT OUTPUTS:
;   122          0183  1 !
;   123          0184  1 !       NONE
;   124          0185  1 !
;   125          0186  1 ! ROUTINE VALUE:
;   126          0187  1 !
;   127          0188  1 !       NONE
;   128          0189  1 !
;   129          0190  1 ! SIDE EFFECTS:
;   130          0191  1 !
;   131          0192  1 !       If lazy-lookahead is in progress, it is resolved.  If an error
;   132          0193  1 !       occurs while resolving lazy-lookahead, PAS$STATUS returns the
;   133          0194  1 !       status of that error.
;   134          0195  1 !
;   135          0196  1 ! SIGNALLED ERRORS:
;   136          0197  1 !
```

```
;  137     0198  1 !       NONE
;  138     0199  1 !
;  139     0200  1 !--
;  140     0201  1
;  141     0202  2     BEGIN
;  142     0203  2
;  143     0204  2     LOCAL
;  144     0205  2         FCB: REF $PAS$FCB_CONTROL_BLOCK,        ! File Control Block
;  145     0206  2         STATUS,                                 ! Function result
;  146     0207  2         PFV_ADDR: VOLATILE,                     ! Enable argument
;  147     0208  2         UNWIND_ACT: VOLATILE,                   ! Enable argument
;  148     0209  2         ERROR_ADDR: VOLATILE;                   ! Enable argument
;  149     0210  2
;  150     0211  2     !+
;  151     0212  2     ! Enable our local error handler.
;  152     0213  2     !-
;  153     0214  2
;  154     0215  2     ENABLE
;  155     0216  2         STATUS_HANDLER (PFV_ADDR, UNWIND_ACT, ERROR_ADDR);
;  156     0217  2
;  157     0218  2     PFV_ADDR = PFV [PFV$R_PFV]; ! Set enable argument
;  158     0219  2
;  159     0220  2     !+
;  160     0221  2     ! Validate PFV and get FCB.
;  161     0222  2     !-
;  162     0223  2
;  163     0224  2     PAS$$VALIDATE_PFV (PFV [PFV$R_PFV]; FCB);
;  164     0225  2
;  165     0226  2     UNWIND_ACT = PAS$K_UNWIND_UNLOCK;           ! Unlock file on unwind
;  166     0227  2
;  167     0228  2     !+
;  168     0229  2     ! Resolve lazy lookahead if necessary.  If an error occurs, our
;  169     0230  2     ! handler will return to our caller with the error code.
;  170     0231  2     !-
;  171     0232  2
;  172     0233  2     IF NOT .PFV [PFV$V_VALID]
;  173     0234  2     THEN
;  174     0235  2         PAS$$LOOK_AHEAD (PFV [PFV$R_PFV], FCB [FCB$R_FCB]; FCB);
;  175     0236  2
;  176     0237  2     !+
;  177     0238  2     ! If file is not open, get the status from the FCB pointer in the PFV
;  178     0239  2     ! which has as a synonmym the field PFV$L_STATUS.
;  179     0240  2     ! It is stored there for errors which occur when the file is not
;  180     0241  2     ! open.
;  181     0242  2     !-
;  182     0243  2
;  183     0244  2     IF NOT .PFV [PFV$V_OPEN]
;  184     0245  2     THEN
;  185     0246  2         STATUS = .PFV [PFV$L_STATUS]
;  186     0247  2     ELSE
;  187     0248  3         BEGIN
;  188     0249  3
;  189     0250  3         !+
;  190     0251  3         ! The file is open.  Look at the status in the FCB.  If zero and the
;  191     0252  3         ! file is at EOF, return -1.  Otherwise, return the status from the FCB.
;  192     0253  3         !-
;  193     0254  3
```

```
;  194        0255  3            STATUS = .FCB [FCB$L_STATUS];
;  195        0256  4            IF (.STATUS EQL 0) AND (.FCB [FCB$V_EOF])
;  196        0257  3            THEN
;  197        0258  3                STATUS = -1;
;  198        0259  2            END;
;  199        0260  2
;  200        0261  2        PFV [PFV$V_LOCK] = 0;                          ! Unlock file
;  201        0262  2        RETURN .STATUS;                                ! Return status value
;  202        0263  2
;  203        0264  1        END;                                           ! End of routine PAS$STATUS


                                                         .TITLE  PAS$STATUS Get status of last I/O operation
                                                         .IDENT  \1-001\

                                                         .EXTRN  PAS$STATUS, PAS$$VALIDATE_PFV
                                                         .EXTRN  PAS$$LOOK_AHEAD

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

                                 00CC 00000              .ENTRY  PAS$STATUS, Save R2,R3,R6,R7          ; 0142
                     5E            08   C2 00002         SUBL2   #8, SP                                ;
                                   7E   D4 00005         CL._    ERROR_ADDR                            ; 0202
                          04       AE   7C 00007         CLRQ    UNWIND_ACT                            ;
                     6D         003C    CF DE 0000A      MOVAL   4$, (FP)                              ;
                     56           04    AC D0 0000F      MOVL    PFV, R6                               ; 0218
              08     AE                 56 D0 00013      MOVL    R6, PFV_ADDR                          ;
                          00000000G     00 16 00017      JSB     PAS$$VALIDATE_PFV                     ; 0224
              04     AE                 01 D0 0001D      MOVL    #1, UNWIND_ACT                        ; 0226
                     06           06    A6 E8 00021      BLBS    6(R6), 1$                             ; 0233
                          00000000G     00 16 00025      JSB     PAS$$LOOK_AHEAD                       ; 0235
       06     07     A6                 05 E0 0002B 1$:  BBS     #5, 7(R6), 2$                         ; 0244
                     50           0C    A6 D0 00030      MOVL    12(R6), STATUS                        ; 0246
                                        0E 11 00034      BRB     3$                                    ;
                     50           D4    A7 D0 00036 2$:  MOVL    -44(FCB), STATUS                      ; 0255
                                        08 12 0003A      BNEQ    3$                                    ; 0256
       03     FD     A7                 05 E1 0003C      BBC     #5, -3(FCB), 3$                       ;
                     50                 01 CE 00041      MNEGL   #1, STATUS                            ; 0258
              07     A6           80    8F 8A 00044 3$:  BICB2   #128, 7(R6)                           ; 0261
                                           04 00049      RET                                           ; 0264
                                      0000 0004A 4$:     .WORD   Save nothing                          ; 0202
                     50           08    AC D0 0004C      MOVL    8(AP), R0                             ;
                     50           04    A0 D0 00050      MOVL    4(R0), R0                             ;
                                  F4    A0 9F 00054      PUSHAB  ERROR_ADDR                            ;
                                  F8    A0 9F 00057      PUSHAB  UNWIND_ACT                            ;
                                  FC    A0 9F 0005A      PUSHAB  PFV_ADDR                              ;
                                        03 DD 0005D      PUSHL   #3                                    ;
                                        5E DD 0005F      PUSHL   SP                                    ;
                     7E           04    AC 7D 00061      MOVQ    4(AP), -(SP)                          ;
              0000V  CF                 03 FB 00065      CALLS   #3, STATUS_HANDLER                    ;
                                           04 0006A      RET                                           ;

; Routine Size:  107 bytes,    Routine Base:  _PAS$CODE + 0000


;  204        0265  1
;  205        0266  1 !<BLF/PAGE>
```

```
;  207     0267  1 %SBTTL 'STATUS_HANDLER - Local handler'
;  208     0268  1 ROUTINE STATUS_HANDLER (
;  209     0269  1         SIGNAL_ARGS: REF BLOCK [, BYTE],        ! Signal arguments array
;  210     0270  1         MECH_ARGS: REF BLOCK [, BYTE],          ! Mechanism arguments array
;  211     0271  1         ENABLE_ARGS: REF VECTOR [, LONG]        ! Enable arguments array
;  212     0272  1     ) =
;  213     0273  1
;  214     0274  1 !++
;  215     0275  1 ! FUNCTIONAL DESCRIPTION:
;  216     0276  1 !
;  217     0277  1 !       This is the condition handler enabled PAS$STATUS.  If the current
;  218     0278  1 !       signal is a Pascal error on the file our establisher was called
;  219     0279  1 !       with, we unwind to the caller of PAS$STATUS with R0 being the
;  220     0280  1 !       status code of the error.
;  221     0281  1 !
;  222     0282  1 !       Upon an unwind, the file variable is unlocked.
;  223     0283  1 !
;  224     0284  1 ! CALLING SEQUENCE:
;  225     0285  1 !
;  226     0286  1 !       status.wlc.v = STATUS_HANDLER (SIGNAL_ARGS.rl.ra, MECH_ARGS.rl.ra,
;  227     0287  1 !                                      ENABLE_ARGS.rl.ra)
;  228     0288  1 !
;  229     0289  1 ! FORMAL PARAMETERS:
;  230     0290  1 !
;  231     0291  1 !       SIGNAL_ARGS     - The signal argument list.
;  232     0292  1 !
;  233     0293  1 !       MECH_ARGS       - The mechanism argument list.
;  234     0294  1 !
;  235     0295  1 !       ENABLE_ARGS     - An array with the following
;  236     0296  1 !                         format:
;  237     0297  1 !
;  238     0298  1 !                               +-----------------+
;  239     0299  1 !                               |   ENB_COUNT     |  <-- ENABLE_ARGS
;  240     0300  1 !                               +-----------------+
;  241     0301  1 !                               |  ENB_PFV ADDR   |
;  242     0302  1 !                               +-----------------+
;  243     0303  1 !                               | ENB_UNWIND_ACT  |
;  244     0304  1 !                               +-----------------+
;  245     0305  1 !                               | ENB_ERROR_ADDR  |
;  246     0306  1 !                               +-----------------+
;  247     0307  1 !
;  248     0308  1 !                         ENB_COUNT is the count of following enable arguments.
;  249     0309  1 !                         The count is always at least 2.
;  250     0310  1 !
;  251     0311  1 !                         ENB_PFV_ADDR - If non-zero, the address of a longword
;  252     0312  1 !                         containing the PFV our establisher is operating on.
;  253     0313  1 !
;  254     0314  1 !                         ENB_UNWIND_ACT - Specifies the action
;  255     0315  1 !                         to take on an unwind.  The values are:
;  256     0316  1 !                          PAS$K_UNWIND_NOP - Do nothing
;  257     0317  1 !                          PAS$K_UNWIND_UNLOCK - Unlock PFV
;  258     0318  1 !
;  259     0319  1 !                         ENB_ERROR_ADDR - Ignored here.
;  260     0320  1 !
;  261     0321  1 ! IMPLICIT INPUTS:
;  262     0322  1 !
;  263     0323  1 !       The signaller's PFV placed as the first FAO argument in the primary
```

```
;  264     0324  1 !       signalled message.
;  265     0325  1 !
;  266     0326  1 ! IMPLICIT OUTPUTS:
;  267     0327  1 !
;  268     0328  1 !       May clear PFV$V_LOCK upon unwind.
;  269     0329  1 !
;  270     0330  1 ! ROUTINE VALUE:
;  271     0331  1 !
;  272     0332  1 !       SS$_RESIGNAL
;  273     0333  1 !
;  274     0334  1 ! SIDE EFFECTS:
;  275     0335  1 !
;  276     0336  1 !       May cause an unwind.
;  277     0337  1 !
;  278     0338  1 !--
;  279     0339  1
;  280     0340  2     BEGIN
;  281     0341  2
;  282     0342  2     LITERAL
;  283     0343  2         ENB_COUNT = 0,          ! Count of enable arguments
;  284     0344  2         ENB_PFV_ADDR = 1,       ! Address of address of PFV
;  285     0345  2         ENB_UNWIND_ACT= 2,      ! Address of unwind action
;  286     0346  2         ENB_ERROR_ADDR = 3;     ! Address of address of unwind PC
;  287     0347  2
;  288     0348  2     !+
;  289     0349  2     ! Determine if this is an unwind.
;  290     0350  2     !-
;  291     0351  2
;  292     0352  2     IF .SIGNAL_ARGS [CHF$L_SIG_NAME] EQLU SS$_UNWIND
;  293     0353  2     THEN
;  294     0354  3         BEGIN
;  295     0355  3         IF ..ENABLE_ARGS [ENB_UNWIND_ACT] NEQ PAS$K_UNWIND_NOP
;  296     0356  3         THEN
;  297     0357  4             BEGIN
;  298     0358  4
;  299     0359  4             !+
;  300     0360  4             ! Get our establisher's PFV.
;  301     0361  4             !-
;  302     0362  4
;  303     0363  4             LOCAL
;  304     0364  4                 PFV: REF $PAS$PFV_FILE_VARIABLE;
;  305     0365  4             PFV = ..ENABLE_ARGS [ENB_PFV_ADDR]; ! Get PFV address
;  306     0366  4
;  307     0367  4             PFV [PFV$V_LOCK] = 0;       ! Unlock PFV
;  308     0368  3             END;
;  309     0369  3         END
;  310     0370  3
;  311     0371  2     ELSE
;  312     0372  2
;  313     0373  3         BEGIN
;  314     0374  3
;  315     0375  3         LOCAL
;  316     0376  3             COND_NAME: BLOCK [4, BYTE], ! Primary condition name
;  317     0377  3             COND_CODE;                  ! Sequence number of error
;  318     0378  3
;  319     0379  3         !+
;  320     0380  3         ! Get primary condition name.
```

```
;  321     0381  3          !-
;  322     0382  3
;  323     0383  3          COND_NAME = .SIGNAL_ARGS [CHF$L_SIG_NAME];
;  324     0384  3
;  325     0385  3          !+
;  326     0386  3          ! Is this a PAS$ error?  If not, resignal.
;  327     0387  3          !-
;  328     0388  3
;  329     0389  3          IF .COND_NAME [STS$V_FAC_NO] NEQU PAS$_FACILITY
;  330     0390  3          THEN
;  331     0391  3              RETURN SS$_RESIGNAL;
;  332     0392  3
;  333     0393  3          !+
;  334     0394  3          ! See if the error message is one which is "trapped"
;  335     0395  3          ! by ERROR:=CONTINUE.  This is done by comparing the
;  336     0396  3          ! message number against a select range.  These are also
;  337     0397  3          ! the messages which STATUS will work on.
;  338     0398  3          !-
;  339     0399  3
;  340     0400  3          COND_CODE = .COND_NAME [STS$V_CODE];     ! Get error number
;  341     0401  3          IF .COND_CODE GEQU PAS$$K_MSGCONTLO AND          ! Lowest number
;  342     0402  3             .COND_CODE LEQU PAS$$K_MSGCONTHI     ! Highest number
;  343     0403  3          THEN
;  344     0404  4              BEGIN
;  345     0405  4
;  346     0406  4              !+
;  347     0407  4              ! See if the PFVs match.  The signaller's PFV is the
;  348     0408  4              ! first FAO parameter in the primary message.
;  349     0409  4              !-
;  350     0410  4
;  351     0411  4              IF .SIGNAL_ARGS [12,0,32,0] EQLA ..ENABLE_ARGS [ENB_PFV_ADDR]
;  352     0412  4              THEN
;  353     0413  4                  !+
;  354     0414  4                  ! We want to return to the caller of PAS$STATUS with the
;  355     0415  4                  ! error number.
;  356     0416  4                  !
;  357     0417  4                  ! Place it in the mechanism args list and do a default
;  358     0418  4                  ! unwind.
;  359     0419  4                  !-
;  360     0420  4
;  361     0421  5                  BEGIN
;  362     0422  5                  MECH_ARGS [CHF$L_MCH_SAVR0] = .COND_CODE - PAS$$K_MSGCONTLO;
;  363     0423  6                  IF NOT $UNWIND ()
;  364     0424  5                  THEN
;  365     0425  5                      $PAS$BUGCHECK (BUG_UNWINDFAIL);
;  366     0426  4                  END;
;  367     0427  3              END;
;  368     0428  2          END;
;  369     0429  2
;  370     0430  2      RETURN SS$_RESIGNAL;                        ! Resignal error
;  371     0431  2
;  372     0432  1      END;                                        ! End of routine STATUS_HANDLER


                                                                   .EXTRN  PAS$_FACILITY, PAS$$K_MSGCONTLO
                                                                   .EXTRN  PAS$$K_MSGCONTHI
```

```
                                                              .EXTRN  SYS$UNWIND, PAS$$BUGCHECK

                                          000C 00000 STATUS_HANDLER:
                                                              .WORD   Save R2,R3                                   ; 0268
                      53    0000G   8F   3C 00002             MOVZWL  #PAS$$K_MSGCONTLO, R3
                      51       04   AC   D0 00007             MOVL    SIGNAL_ARGS, R1                              ; 0352
          00000920    8F       04   A1   D1 0000B             CMPL    4(R1), #2336
                                    14   12 00013             BNEQ    1$
                      50       0C   AC   D0 00015             MOVL    ENABLE_ARGS, R0                              ; 0355
                               08   B0   D5 00019             TSTL    @8(R0)
                                    54   13 0001C             BEQL    2$
                      50       04   B0   D0 0001E             MOVL    @4(R0), PFV                                  ; 0365
                07    A0       80   8F   8A 00022             BICB2   #128, 7(PFV)                                 ; 0367
                                    49   11 00027             BRB     2$                                           ; 0352
                      50       04   A1   D0 00029 1$:         MOVL    4(R1), COND_NAME                             ; 0383
     00G         50   0C             10   ED 0002D             CMPZV   #16, #12, COND_NAME, S^PAS$_FACILITY         ; 0389
                                    3E   12 00032             BNEQ    2$
     52          50   0C             03   EF 00034             EXTZV   #3, #12, COND_NAME, COND_CODE                ; 0400
                      53             52   D1 00039             CMPL    COND_CODE, R3                                ; 0401
                                    34   1F 0003C             BLSSU   2$
          00000000G   8F             52   D1 0003E             CMPL    COND_CODE, #PAS$$K_MSGCONTHI                 ; 0402
                                    2B   1A 00045             BGTRU   2$
                      50       0C   AC   D0 00047             MOVL    ENABLE_ARGS, R0                              ; 0411
                04    B0       0C   A1   D1 0004B             CMPL    12(R1), @4(R0)
                                    20   12 00050             BNEQ    2$
                      51       08   AC   D0 00052             MOVL    MECH_ARGS, R1                                ; 0422
           0C    A1   52             53   C3 00056             SUBL3   R3, COND_CODE, 12(R1)
                                    7E   7C 0005B             CLRQ    -(SP)                                        ; 0423
          00000000G   00             02   FB 0005D             CALLS   #2, SYS$UNWIND
                      0B             50   E8 00064             BLBS    R0, 2$
                                    03   DD 00067             PUSHL   #3                                           ; 0425
          00000000G   00             01   FB 00069             CALLS   #1, PAS$$BUGCHECK
                                    06   11 00070             BRB     3$
                      50     0918   8F   3C 00072 2$:         MOVZWL  #2328, R0                                    ; 0430
                                         04 00077             RET
                                    50   D4 00078 3$:         CLRL    R0                                           ; 0432
                                         04 0007A             RET

; Routine Size:  123 bytes,    Routine Base:  _PAS$CODE + 006B


;  373          0433  1
;  374          0434  1 !<BLF/PAGE>
```

```
;  376          0435  1 END                                   ! End of module PAS$STATUS
;  377          0436  1
;  378          0437  0 ELUDOM
```

PSECT SUMMARY

Name	Bytes	Attributes
_PAS$CODE	230	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	8	0	581	00:01.0
_255DUA28:[PASRTL.OBJ]PASLIB.L32;1	427	91	21	33	00:00.4

COMMAND QUALIFIERS

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

; Size:         230 code + 0 data bytes
; Run Time:       00:07.2
; Elapsed Time:   00:26.2
; Lines/CPU Min:   3667
; Lexemes/CPU-Min: 10867
; Memory Used:  75 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