```
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    GGGGGGGG    OOOOOO    TTTTTTTTTT    OOOOOO
PPPPPPPP      AAAAAA      SSSSSSSS    GGGGGGGG    OOOOOO    TTTTTTTTTT    OOOOOO
PP      PP   AA    AA   SS          GG          OO    OO      TT      OO    OO
PP      PP   AA    AA   SS          GG          OO    OO      TT      OO    OO
PP      PP   AA    AA   SS          GG          OO    OO      TT      OO    OO
PP      PP   AA    AA   SS          GG          OO    OO      TT      OO    OO
PPPPPPPP     AA    AA     SSSSSS    GG          OO    OO      TT      OO    OO
PPPPPPPP     AA    AA     SSSSSS    GG          OO    OO      TT      OO    OO
PP           AAAAAAAAAA         SS  GG  GGGGGG  OO    OO      TT      OO    OO
PP           AAAAAAAAAA         SS  GG  GGGGGG  OO    OO      TT      OO    OO
PP           AA    AA           SS  GG      GG  OO    OO      TT      OO    OO      ....
PP           AA    AA           SS  GG      GG  OO    OO      TT      OO    OO      ....
PP           AA    AA   SSSSSSSS    GGGGGG      OOOOOO        TT        OOOOOO      ....
PP           AA    AA   SSSSSSSS    GGGGGG      OOOOOO        TT        OOOOOO      ....

LL             IIIIII      SSSSSSSS
LL             IIIIII      SSSSSSSS
LL               II      SS
LL               II      SS
LL               II      SS
LL               II      SS
LL               II        SSSSSS
LL               II        SSSSSS
LL               II              SS
LL               II              SS
LL               II              SS
LL               II              SS
LLLLLLLLLL     IIIIII    SSSSSSSS
LLLLLLLLLL     IIIIII    SSSSSSSS
```

Table of contents

```
0000     1          .TITLE  PAS$GOTO - Perform up-level GOTO
0000     2          .IDENT  /2-001/                 ; File: PASGOTO.MAR Edit: SBL2001
0000     3
0000     4 ;
0000     5 ;*****************************************************************************
0000     6 ;*                                                                           *
0000     7 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     8 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     9 ;*  ALL RIGHTS RESERVED.                                                     *
0000    10 ;*                                                                           *
0000    11 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    12 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    13 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    14 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    15 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    16 ;*  TRANSFERRED.                                                             *
0000    17 ;*                                                                           *
0000    18 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    19 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    20 ;*  CORPORATION.                                                             *
0000    21 ;*                                                                           *
0000    22 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    23 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    24 ;*                                                                           *
0000    25 ;*                                                                           *
0000    26 ;*****************************************************************************
0000    27 ;
0000    28
0000    29 ;++
0000    30 ; FACILITY: VAX-11 PASCAL Language Support
0000    31 ;
0000    32 ; ABSTRACT:
0000    33 ;
0000    34 ;       This module contains PAS$GOTO, which performs an up-level GOTO
0000    35 ;       for Pascal routines.
0000    36 ;
0000    37 ; ENVIRONMENT: User mode, AST Reentrant
0000    38 ;
0000    39 ; AUTHOR: Steven B. Lionel, CREATION DATE: 28-Jan-1983
0000    40 ;         Special thanks to Bevin Brett.
0000    41 ;
0000    42 ; MODIFIED BY:
0000    43 ;
0000    44 ; 2-001 - Complete rewrite of orginal BLISS version which did not work
0000    45 ;         when called from condition handlers.  SBL 28-Jan-1983
0000    46 ;--
```

```
                        .SBTTL  DECLARATIONS
                ;
                ; LIBRARY MACRO CALLS:
                ;

                        $CHFDEF                 ; Condition Handling symbols
                        $SSDEF                  ; SS$_ symbols

                ;
                ; EXTERNAL DECLARATIONS:
                ;

                        .DSABL  GBL             ; Force all external symbols to be declared
                        .EXTRN  LIB$STOP        ; Signal non-continuable exception
                        .EXTRN  PAS$_GOTO       ; Up-level GOTO
                        .EXTRN  PAS$_GOTOFAILED ; Up-level GOTO failed
                        .EXTRN  SYS$UNWIND      ; $UNWIND system service

                ;
                ; MACROS:
                ;
                ;       NONE
                ;
                ; EQUATED SYMBOLS:
                ;
                ;       NONE
                ;
                ; OWN STORAGE:
                ;
                ;       NONE
                ;
                ; PSECT DECLARATIONS:
                ;
00000000                .PSECT _PAS$CODE PIC, USR, CON, REL, LCL, SHR, -
                                EXE, RD, NOWRT, LONG
```

```
                     0000    85          .SBTTL  PAS$GOTO - Perform up-level GOTO
                     0000    86 ;++
                     0000    87 ; FUNCTIONAL DESCRIPTION:
                     0000    88 ;
                     0000    89 ;       This procedure is called by PASCAL compiled code to perform
                     0000    90 ;       an up-level GOTO.  Functionally, it performs a $UNWIND to
                     0000    91 ;       the specified frame and PC.  The actual implementation is
                     0000    92 ;       described in detail below.
                     0000    93 ;
                     0000    94 ; CALLING SEQUENCE:
                     0000    95 ;
                     0000    96 ;       CALL PAS$GOTO (dest_FP.ra.v, dest_PC.jzi.r)
                     0000    97 ;
                     0000    98 ; FORMAL PARAMETERS:
                     0000    99 ;
00000004             0000   100          dest_FP = 4             ; The FP of the destination frame
                     0000   101                                  ; If the signal is PAS$_GOTO, it has two
                     0000   102                                  ; "FAO arguments", the destination FP and PC.
                     0000   103
00000008             0000   104          dest_PC = 8             ; The PC of the destination instruction
                     0000   105
                     0000   106 ;
                     0000   107 ; IMPLICIT INPUTS:
                     0000   108 ;
                     0000   109 ;       NONE
                     0000   110 ;
                     0000   111 ; IMPLICIT OUTPUTS:
                     0000   112 ;
                     0000   113 ;       NONE
                     0000   114 ;
                     0000   115 ; COMPLETION STATUS:
                     0000   116 ;
                     0000   117 ;       NONE
                     0000   118 ;
                     0000   119 ; SIDE EFFECTS:
                     0000   120 ;
                     0000   121 ;       Functionally performs a $UNWIND to the specified FP and PC.
                     0000   122 ;
                     0000   123 ;--
```

```
                        ;+
                        ; Implementation notes:
                        ;
                        ;       An "up-level" GOTO is a GOTO where the destination is not in the
                        ;       same stack frame (procedure incarnation) as the origination.  Ideally,
                        ;       what one wants to do is unwind the stack frames back to the
                        ;       desired frame, and then begin executing at the destination labelled
                        ;       instruction.  The unwind is necessary to restore saved registers;
                        ;       one can't simply JMP to the instruction since the stack frame and
                        ;       register contents would be inconsistent.
                        ;
                        ;       There is, of course, the system service $UNWIND that seems to do
                        ;       exactly what we want.  You specify to $UNWIND the number of frames
                        ;       to remove and the desired PC, and off it goes.  The first problem
                        ;       with this is that you can only unwind while in a condition handler
                        ;       (or in a procedure called from a condition handler).  This is not
                        ;       much of a problem, one can simply signal a special exception and
                        ;       intercept it in a handler, which then does the $UNWIND.  The
                        ;       second problem is that, while $UNWIND wants a number of frames to
                        ;       remove, we don't know how many frames distant we are from the
                        ;       destination; we do know the FP value of the destination frame.  So,
                        ;       the initial implementation searched through the stack frame chain,
                        ;       counting until it found the desired FP.  It then signalled PAS$_GOTO
                        ;       with arguments of the count and PC, and its own handler did the
                        ;       unwind.  This worked well in normal cases, but failed spectacularly
                        ;       if it was called from a condition handler.
                        ;
                        ;       The problem was simply that when $UNWIND counts stack frames, and
                        ;       it comes across a condition handler, it "skips" to the establisher's
                        ;       frame without counting intervening frames.  This is correct according
                        ;       to the handler search algorithm.  Because PAS$GOTO wasn't taking this
                        ;       into account, the number of frames to unwind that it specified was
                        ;       wrong.
                        ;
                        ;       It is difficult, though possible, to have PAS$GOTO count frames in the
                        ;       same manner as $UNWIND.  If this is done, one finds another problem;
                        ;       this time due to a design flaw in $UNWIND.  Basically, if one
                        ;       specifies a non-zero number of frames to unwind, with the intention
                        ;       of unwinding to an establisher's frame, $UNWIND removes one stack frame
                        ;       too many.  If you decrease the count by one, you unwind only to
                        ;       the handler.  Thus, it is impossible to unwind exactly to an establisher
                        ;       frame if that signal is not the current one.  Since being able to
                        ;       GOTO elsewhere in the establisher's frame is a desireable feature, this
                        ;       is unacceptable.
                        ;
                        ;       An intermediate implementation was tried which simply establishes
                        ;       a handler in the destination frame, signals PAS$_GOTO, and lets
                        ;       that handler unwind to its establisher.  This doesn't work when there
                        ;       is already a signal in progress since the special GOTO handler is
                        ;       skipped.
                        ;
                        ;       The successful solution is somewhat complicated, and is actually
                        ;       two solutions in one.  There are two interesting cases of an
                        ;       up-level GOTO:
                        ;               1. There is no signal currently in progress
                        ;               2. There are one or more signals currently in progress
                        ;
```

```
0000   182 ;       The first case can be solved with either the original method or
0000   183 ;       with the "intermediate implementation" where a special handler
0000   184 ;       is temporarily established in the destination frame.  The latter
0000   185 ;       is what is used; PAS$HANDLER, if already established, serves as
0000   186 ;       that special handler, or PAS$$UNWIND_GOTO is established if there
0000   187 ;       is no handler.  We assume that no handler other than PAS$HANDLER
0000   188 ;       is established in the destination frame.  This is reasonable, because
0000   189 ;       the only way it could get there is by the user calling LIB$ESTABLISH,
0000   190 ;       and if this was done, the user got a compile-time warning from
0000   191 ;       VAX-11 PASCAL saying that LIB$ESTABLISH was incompatible with
0000   192 ;       VAX-11 PASCAL.
0000   193 ;
0000   194 ;       Once a handler is established in the destination frame, PAS$_GOTO
0000   195 ;       is signalled, with two FAO parameters of the destination FP and
0000   196 ;       PC.  Note that the stack-frame search of the original implementation
0000   197 ;       is no longer present.  If no other exception is in progress, this
0000   198 ;       signal will be caught by the handler in the destination frame, which
0000   199 ;       will then unwind zero frames to the establisher at the destination PC.
0000   200 ;
0000   201 ;       The more interesting case is when there is an exception in progress.
0000   202 ;       PAS$HANDLER, which was established when the user used the ESTABLISH
0000   203 ;       builtin, and which has already been called for the current signal,
0000   204 ;       has itself established a handler PAS$$GOTO_HANDLER.  This handler
0000   205 ;       causes an unwind back to the frame of its establisher (PAS$HANDLER),
0000   206 ;       but at PC UNWIND_TO_ESTABLISHER.  This effectively removes the last
0000   207 ;       exception (PAS$_GOTO).  Before unwinding, the destination FP and PC
0000   208 ;       are loaded into the saved R0 and R1 so that they can be communicated
0000   209 ;       to UNWIND_TO_ESTABLISHER.
0000   210 ;
0000   211 ;       UNWIND_TO_ESTABLISHER then unwinds zero frames to the establisher, but
0000   212 ;       at PC JUMP_TO_DEST.  Again, R0 and R1 have the saved FP and PC.
0000   213 ;       JUMP_TO_DEST looks to see if the destination FP is the same as its
0000   214 ;       current FP, which it might not be.  If it is, then it simply jumps
0000   215 ;       to the destination PC.  Otherwise, it calls PAS$GOTO again with
0000   216 ;       the original arguments.  Eventually, all signals between the source
0000   217 ;       and the destination of the GOTO will be unwound.
0000   218 ;
0000   219 ;       The following problems with $UNWIND are known:
0000   220 ;           1.  You can't unwind more than one exception reliably.
0000   221 ;           2.  Unwinding zero frames leaves the signal and mechanism arglists,
0000   222 ;               along with some other stuff, on the stack.  This doesn't
0000   223 ;               bother us as PASCAL always readjusts the stack at GOTO
0000   224 ;               destinations.
0000   225 ;           3.  Unwinding zero frames doesn't restore the saved R0 and R1
0000   226 ;               from the mchargs list.  This is solved by manually loading
0000   227 ;               the registers before doing the RET from the handler.
0000   228 ;
0000   229 ;-
```

```
                     0000  0000    231          .ENTRY  PAS$GOTO, ^M<>
                           0002    232
                           0002    233 ;+
                           0002    234 ; Look in the destination frame to see if there is a handler.  If so,
                           0002    235 ; we assume that it is PAS$HANDLER and do nothing.  If not, we establish
                           0002    236 ; PAS$$UNWIND_GOTO there.  PAS$HANDLER itself establishes PAS$$GOTO_HANDLER.
                           0002    237 ; One of these two handlers will catch the signal of PAS$_GOTO we will make.
                           0002    238 ;-
                           0002    239
            7E   04 AC     7D  0002    240          MOVQ    dest_FP(AP), -(SP)      ; Push destination FP and PC
                 00 BE     D5  0006    241          TSTL    @(SP)                   ; Does destination frame have a handler?
                       06  12  0009    242          BNEQ    10$                     ; Skip if it does
          00 BE    008E'CF   9E  000B    243          MOVAB   W^PAS$$UNWIND_GOTO, @(SP) ; Establish PAS$$UNWIND_GOTO
                           0011    244
                           0011    245 ;+
                           0011    246 ; Now signal PAS$_GOTO with FAO arguments of the destination FP and PC.  This
                           0011    247 ; will be intercepted by PAS$$GOTO_HANDLER or PAS$$UNWIND_GOTO
                           0011    248 ; to actually do the unwinds.
                           0011    249 ;-
                           0011    250
                       02  DD  0011    251 10$:     PUSHL   #2                      ; Two arguments already pushed
              00000000'8F  DD  0013    252          PUSHL   #PAS$_GOTO              ; "Up-level GOTO"
          00000000'GF   04  FB  0019    253          CALLS   #4, G^LIB$STOP          ; Signal it
                           0020    254                                          ; Can never return from LIB$STOP
                           0020    255
```

```
                     0020   257          .SBTTL  PAS$$GOTO_HANDLER - Established by PAS$HANDLER
                     0020   258 ;++
                     0020   259 ; FUNCTIONAL DESCRIPTION:
                     0020   260 ;
                     0020   261 ;       This is a condition handler established by PAS$HANDLER which
                     0020   262 ;       intercepts PAS$_GOTO exceptions and unwinds back to its
                     0020   263 ;       establisher's frame (PAS$HANDLER) but at PC UNWIND_TO_ESTABLISHER.
                     0020   264 ;
                     0020   265 ; CALLING SEQUENCE:
                     0020   266 ;
                     0020   267 ;       ret-status = PAS$$GOTO_HANDLER (sigargs.rlu.r, mchargs.rlu.r)
                     0020   268 ;
                     0020   269 ; FORMAL PARAMETERS:
                     0020   270 ;
                     0020   271
00000004             0020   272          sigargs = 4             ; The signal arguments list
                     0020   273                                  ; If the signal is PAS$_GOTO, it has two
                     0020   274                                  ; "FAO arguments", the destination FP and PC.
                     0020   275
00000008             0020   276          mchargs = 8             ; The mechanism arguments list
                     0020   277
                     0020   278 ;
                     0020   279 ; IMPLICIT INPUTS:
                     0020   280 ;
                     0020   281 ;       NONE
                     0020   282 ;
                     0020   283 ; IMPLICIT OUTPUTS:
                     0020   284 ;
                     0020   285 ;       NONE
                     0020   286 ;
                     0020   287 ; COMPLETION STATUS:
                     0020   288 ;
                     0020   289 ;       NONE
                     0020   290 ;
                     0020   291 ; SIDE EFFECTS:
                     0020   292 ;
                     0020   293 ;       Unwinds back to its establisher (PAS$HANDLER), but at PC
                     0020   294 ;       UNWIND_TO_ESTABLISHER.
                     0020   295 ;
                     0020   296 ;--
                     0020   297
                0000 0020   298          .ENTRY  PAS$$GOTO_HANDLER, ^M<>
           51   04 AC   D0  0022   299          MOVL    sigargs(AP), R1                 ; Get signal arguments list
00000000'8F   04 A1   D1  0026   300          CMPL    CHF$L_SIG_NAME(R1), #PAS$_GOTO  ; Is this PAS$_GOTO?
                 06   13  002E   301          BEQL    10$                             ; If so, keep going
         50 0918 8F   3C  0030   302          MOVZWL  #SS$_RESIGNAL, R0               ; Resignal this exception
                 04   04  0035   303          RET                                     ; Return to CHF
                      0036   304
                      0036   305 ;+
                      0036   306 ; Unwind the stack frames back to our establisher, PAS$$HANDLER.  Use the
                      0036   307 ; saved R0 and R1 to communicate the destination FP and PC.
                      0036   308 ;-
                      0036   309
           50   08 AC   D0  0036   310 10$:     MOVL    mchargs(AP), R0                 ; Get mechanism arguments list
        0C A0   0C A1   7D  003A   311          MOVQ    12(R1), CHF$L_MCH_SAVR0(R0)     ; Store destination FP and PC
                            003F   312                                                  ; in saved R0 and R1
                61'AF   9F  003F   313          PUSHAB  B^UNWIND_TO_ESTABLISHER         ; Unwind to UNWIND_TO_ESTABLISHER
```

```
                08 A0    9F  0042   314          PUSHAB  CHF$L_MCH_DEPTH(R0)                ; In establisher's frame
00000000'GF     02    FB  0045   315          CALLS   #2, G^SYS$UNWIND                    ; Do the unwind
              01 50   E9  004C   316          BLBC    R0, UNWIND_FAILED                   ; Skip if unwind unsuccessful
                      04  004F   317          RET                                         ; Return to UNWIND service
                          0050   318
                          0050   319 ;+
                          0050   320 ; The $UNWIND was unsuccessful.  Signal PAS$_GOTOFAILED.
                          0050   321 ;-
                          0050   322
                          0050   323 UNWIND_FAILED:
                   50     DD  0050   324          PUSHL   R0                                  ; Unwind failure status
                   7E     D4  0052   325          CLRL    -(SP)                               ; Zero FAO arguments
         00000000'8F      DD  0054   326          PUSHL   #PAS$_GOTOFAILED
00000000'GF     03    FB  005A   327          CALLS   #3, G^LIB$STOP                      ; Signal PAS$_GOTOFAILED
```

```
                                0061   329 ;+
                                0061   330 ; UNWIND_TO_ESTABLISHER - This section of code is unwound to by
                                0061   331 ; PAS$$GOTO_HANDLER.  When we get here, the current frame is that of
                                0061   332 ; PAS$HANDLER, R0 contains the destination frame and R1 the destination PC.
                                0061   333 ; In other words, it is as if we had unwound back to PAS$HANDLER, but at
                                0061   334 ; a different PC.  It is assumed that AP has not been modified.
                                0061   335 ;
                                0061   336 ; An unwind is done of "depth" frames back to the establisher of PAS$HANDLER.
                                0061   337 ; Although the frame will be that of the establisher, the PC will be
                                0061   338 ; our own JUMP_TO_DEST, below.
                                0061   339 ;-
                                0061   340
                                0061   341 UNWIND_TO_ESTABLISHER:
                     7E   50    7D  0061   342         MOVQ    R0, -(SP)                       ; Push dest FP and PC
                  50    08 AC    D0  0064   343         MOVL    mchargs(AP), R0                 ; Get mechanism args list
                  0C A0    6E    7D  0068   344         MOVQ    (SP), CHF$L_MCH_SAVR0(R0)       ; Save dest FP and PC in R0-R1
                        7D'AF    9F  006C   345         PUSHAB  B^JUMP_TO_DEST                  ; Unwind to JUMP_TO_DEST
                     08 A0    9F  006F   346         PUSHAB  CHF$L_MCH_DEPTH(R0)             ; Unwind to establisher
          00000000'GF    02    FB  0072   347         CALLS   #2, G^SYS$UNWIND                ; Do the unwind
                     50    6E    7D  0079   348         MOVQ    (SP), R0                        ; Because we might be unwinding
                                007C   349                                                 ; zero frames, and $UNWIND
                                007C   350                                                 ; currently doesn't restore
                                007C   351                                                 ; R0 and R1 from the mechanism
                                007C   352                                                 ; arguments list, restore them
                                007C   353                                                 ; here.
                           04   007C   354         RET                                     ; Return to JUMP_TO_DEST
                                007D   355
                                007D   356 ;+
                                007D   357 ; JUMP_TO_DEST - We get here by means of the $UNWIND in UNWIND_TO_ESTABLISHER.
                                007D   358 ; The current frame is that of the establisher of the handler that found this
                                007D   359 ; exception, but that is not necessarily our destination.  R0 contains the
                                007D   360 ; destination FP and R1 the destination PC.  If this is the correct frame,
                                007D   361 ; just JMP to the destination PC.  Note that there may be garbage on the
                                007D   362 ; stack left by the CHF - we depend on the PASCAL compiled code to readjust
                                007D   363 ; SP at the destination of GOTOs.
                                007D   364 ;
                                007D   365 ; If this is not the correct FP, simply re-call PAS$GOTO.  Eventually we'll
                                007D   366 ; get to the right frame.
                                007D   367 ;-
                                007D   368
                                007D   369 JUMP_TO_DEST:
                     5D   50    D1  007D   370         CMPL    R0, FP                  ; Is this the destination frame?
                          02    12  0080   371         BNEQ    10$                     ; Skip if not
                          61    17  0082   372         JMP     (R1)                    ; It is - jump to the destination PC
                     7E   50    7D  0084   373 10$:    MOVQ    R0, -(SP)               ; Iteratively call PAS$GOTO
          00000000'GF    02    FB  0087   374         CALLS   #2, G^PAS$GOTO
```

```
                     008E   376          .SBTTL  PAS$$UNWIND_GOTO - Unwind to destination FP and PC
                     008E   377 ;++
                     008E   378 ; FUNCTIONAL DESCRIPTION:
                     008E   379 ;
                     008E   380 ;       This is a condition handler established by PAS$GOTO in the
                     008E   381 ;       destination frame of an up-level GOTO.  It intercepts PAS$_GOTO
                     008E   382 ;       exceptions and initiates an unwind back to the destination
                     008E   383 ;       frame and PC.  This routine is also called by PAS$HANDLER if
                     008E   384 ;       it detects PAS$_GOTO.
                     008E   385 ;
                     008E   386 ; CALLING SEQUENCE:
                     008E   387 ;
                     008E   388 ;       ret-status = PAS$$UNWIND_GOTO (sigargs.rlu.r, mchargs.rlu.r)
                     008E   389 ;
                     008E   390 ; FORMAL PARAMETERS:
                     008E   391 ;
                     008E   392
00000004             008E   393          sigargs = 4             ; The signal arguments list
                     008E   394                                  ; If the signal is PAS$_GOTO, it has two
                     008E   395                                  ; "FAO arguments", the destination FP and PC.
                     008E   396
00000008             008E   397          mchargs = 8             ; The mechanism arguments list
                     008E   398
                     008E   399 ;
                     008E   400 ; IMPLICIT INPUTS:
                     008E   401 ;
                     008E   402 ;       NONE
                     008E   403 ;
                     008E   404 ; IMPLICIT OUTPUTS:
                     008E   405 ;
                     008E   406 ;       NONE
                     008E   407 ;
                     008E   408 ; COMPLETION STATUS:
                     008E   409 ;
                     008E   410 ;       NONE
                     008E   411 ;
                     008E   412 ; SIDE EFFECTS:
                     008E   413 ;
                     008E   414 ;       Unwinds back to the destination frame and PC.
                     008E   415 ;
                     008E   416 ;--
                     008E   417
                0004 008E   418          .ENTRY  PAS$$UNWIND_GOTO, ^M<R2>
          50    04 AC   7D   0090   419          MOVQ    sigargs(AP), R0                 ; Get signal and mechanism lists
00000000'8F     04 A0   D1   0094   420          CMPL    CHF$L_SIG_NAME(R0), #PAS$_GOTO  ; Is this PAS$_GOTO?
                   07   12   009C   421          BNEQ    10$                             ; If not, resignal
       04 A1    0C A0   D1   009E   422          CMPL    12(R0), CHF$L_MCH_FRAME(R1)     ; Is establisher FP the dest FP?
                   06   13   00A3   423          BEQL    20$                             ; Skip if so
        50  0918 8F   3C   00A5   424 10$:       MOVZWL  #SS$_RESIGNAL, R0               ; Resignal this exception
                   04   00AA   425          RET                                     ; Return to CHF
                        00AB   426
                        00AB   427 ;+
                        00AB   428 ; If the handler established in our "establisher's" frame is PAS$$UNWIND_GOTO,
                        00AB   429 ; (which it wouldn't be if we were called from PAS$HANDLER), remove our
                        00AB   430 ; address as that frame's handler.
                        00AB   431 ;-
                        00AB   432
```

```
                                                J 14
                52   E0 AF    9E  00AB    433 20$:    MOVAB   B^PAS$$UNWIND_GOTO, R2          ; Get address of our entry mask
              04 B1    52    D1  00AF    434          CMPL    R2, @CHF$L_MCH_FRAME(R1)       ; Is it the same as establishers
                                   00B3    435                                                 ; handler?
                       03    12  00B3    436          BNEQ    30$                            ; Skip if not
                    04 B1    D4  00B5    437          CLRL    @CHF$L_MCH_FRAME(R1)           ; Cancel the handler
                                   00B8    438
                                   00B8    439 ;+
                                   00B8    440 ; Unwind the stack frames back to our establisher, the destination frame,
                                   00B8    441 ; and to the destination PC.
                                   00B8    442 ;-
                                   00B8    443
                    10 A0    DD  00B8    444 30$:    PUSHL   16(R0)                         ; Push destination PC
                    08 A1    9F  00BB    445          PUSHAB  CHF$L_MCH_DEPTH(R1)            ; Unwind to establisher
        00000000'GF    02    FB  00BE    446          CALLS   #2, G^SYS$UNWIND               ; Do the unwind
                 03 50    E8  00C5    447          BLBS    R0, 40$                        ; Return if successful
                   FF85   31  00C8    448          BRW     UNWIND_FAILED                  ; Signal failure of UNWIND
                          04  00CB    449 40$:    RET                                    ; Return to UNWIND service
                                   00CC    450
                                   00CC    451          .END
```

```
CHF$L_MCH_DEPTH                 = 00000008
CHF$L_MCH_FRAME                 = 00000004
CHF$L_MCH_SAVR0                 = 0000000C
CHF$L_SIG_NAME                  = 00000004
DEST_FP                         = 00000004
JUMP_TO_DEST                      0000007D R      02
LIB$STOP                          ******** *X    00
PAS$$GOTO_HANDLER                 00000020 RG     02
PAS$$UNWIND_GOTO                  0000008E RG     02
PAS$GOTO                          00000000 RG     02
PAS$_GOTO                         ******** *X    00
PAS$_GOTOFAILED                   ******** *X    00
SIGARGS                         = 00000004
SS$_RESIGNAL                    = 00000918
SYS$UNWIND                        ******** *X    00
UNWIND_FAILED                     00000050 R      02
UNWIND_TO_ESTABLISHER             00000061 R      02
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	00000000 (0.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
_PAS$CODE	000000CC (204.)	02 (2.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	10	00:00:00.08	00:00:00.25
Command processing	74	00:00:00.67	00:00:02.60
Pass 1	182	00:00:04.37	00:00:14.87
Symbol table sort	0	00:00:00.63	00:00:01.89
Pass 2	94	00:00:01.19	00:00:07.07
Symbol table output	2	00:00:00.05	00:00:00.47
Psect synopsis output	3	00:00:00.02	00:00:00.02
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	367	00:00:07.01	00:00:27.17

The working set limit was 900 pages.
24301 bytes (48 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 434 non-local and 7 local symbols.
451 source lines were read in Pass 1, producing 19 object records in Pass 2.
9 pages of virtual memory were used to define 8 macros.

```
                                  +----------------------------+
                                  ! Macro library statistics !
                                  +----------------------------+

Macro library name                 Macros defined
------------------                 --------------
_$255$DUA28:[SYSLIB]STARLET.MLB;2              5

486 GETS were required to define 5 macros.

There were no errors, warnings or information messages.

MACRO/ENABLE=SUPPRESSION/DISABLE=(GLOBAL,TRACEBACK)/LIS=LIS$:PASGOTO/OBJ=OBJ$:PASGOTO MSRC$:PASGOTO/UPDATE=(ENH$:PASGOTO)
```

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

PASFVOUTP
LIS

PASEOLN2
LIS

PASHEAP
LIS

PASHANDLE
LIS

PASFAB
LIS

PASGET
LIS

PASCVTRT
LIS

PASDATE
LIS

PASEOF2
LIS

PASFINDK
LIS

PASFVINPU
LIS

PASEXPO
LIS

PASGOTO
LIS

PASFILEUT
LIS

PASHALT
LIS

PASDELETE
LIS

PASFIND2
LIS