G 6

```
PPPPPPPPPPPP        AAAAAAAAA        SSSSSSSSSSSS  RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
PPPPPPPPPPPP        AAAAAAAAA        SSSSSSSSSSSS  RRRRRRRRRRRR     TTTTTTTTTTTTTTT  LLL
PPPPPPPPPPPP        AAAAAAAAA        SSSSSSSSSSSS  RRRRRRRRRRRR     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     RRRRRRRRRRRR           TTT        LLL
PPPPPPPPPPPP     AAA         AAA     SSSSSSSSS     RRRRRRRRRRRR           TTT        LLL
PPPPPPPPPPPP     AAA         AAA     SSSSSSSSS     RRRRRRRRRRRR           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  SSSSSSSSSSSS     RRR         RRR        TTT        LLLLLLLLLLLLLLL
PPP              AAA         AAA  SSSSSSSSSSSS     RRR         RRR        TTT        LLLLLLLLLLLLLLL
PPP              AAA         AAA  SSSSSSSSSSSS     RRR         RRR        TTT        LLLLLLLLLLLLLLL
```

```
_$2

Sym
---


PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS

PAS

PAS

PAS

PAS
PAS
PAS
PAS
PAS

PAS
PAS
PAS
PAS
PAS
PAS
PAS
PAS
```

K 13

```
**FILE**ID**PASGOTO




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
```

```
PA
Sy

CH
CH
CH
CH
DE
JU
LI
PA
PA
PA
PA
PA
SI
SS
SY
UN
UN



PS
--
.
$A
_F



Ph
--
In
Co
Pa
Sy
Pa
Sy
Ps
Cr
As

TM
24
TM
45
9
```

Table of contents

```
PA
VA

Ma
--
_$

48

Th

MA
```

```
0000        .TITLE  PAS$GOTO - Perform up-level GOTO
0000        .IDENT  /2-001/                  ; File: PASGOTO.MAR Edit: SBL2001
0000
0000   ;
0000   ;****************************************************************************
0000   ;*                                                                          *
0000   ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000   ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                  *
0000   ;*  ALL RIGHTS RESERVED.                                                    *
0000   ;*                                                                          *
0000   ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED   *
0000   ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000   ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER   *
0000   ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY   *
0000   ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY   *
0000   ;*  TRANSFERRED.                                                            *
0000   ;*                                                                          *
0000   ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE   *
0000   ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000   ;*  CORPORATION.                                                            *
0000   ;*                                                                          *
0000   ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS   *
0000   ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                 *
0000   ;*                                                                          *
0000   ;*                                                                          *
0000   ;****************************************************************************
0000   ;
0000
0000   ;++
0000   ; FACILITY: VAX-11 PASCAL Language Support
0000   ;
0000   ; ABSTRACT:
0000   ;
0000   ;       This module contains PAS$GOTO, which performs an up-level GOTO
0000   ;       for Pascal routines.
0000   ;
0000   ; ENVIRONMENT: User mode, AST Reentrant
0000   ;
0000   ; AUTHOR: Steven B. Lionel, CREATION DATE: 28-Jan-1983
0000   ;         Special thanks to Bevin Brett.
0000   ;
0000   ; MODIFIED BY:
0000   ;
0000   ; 2-001 - Complete rewrite of orginal BLISS version which did not work
0000   ;         when called from condition handlers.  SBL 28-Jan-1983
0000   ;--
```

```
    0000          .SBTTL  DECLARATIONS
    0000  ;
    0000  ; LIBRARY MACRO CALLS:
    0000  ;
    0000
    0000          $CHFDEF                         ; Condition Handling symbols
    0000          $SSDEF                          ; SS$_ symbols
    0000
    0000  ;
    0000  ; EXTERNAL DECLARATIONS:
    0000  ;
    0000
    0000          .DSABL  GBL                     ; Force all external symbols to be declared
    0000          .EXTRN  LIB$STOP                ; Signal non-continuable exception
    0000          .EXTRN  PAS$_GOTO               ; Up-level GOTO
    0000          .EXTRN  PAS$_GOTOFAILED         ; Up-level GOTO failed
    0000          .EXTRN  SYS$UNWIND              ; $UNWIND system service
    0000
    0000  ;
    0000  ; MACROS:
    0000  ;
    0000          NONE
    0000  ;
    0000  ; EQUATED SYMBOLS:
    0000  ;
    0000          NONE
    0000  ;
    0000  ; OWN STORAGE:
    0000  ;
    0000          NONE
    0000  ;
    0000  ; PSECT DECLARATIONS:
    0000  ;
00000000          .PSECT _PAS$CODE PIC, USR, CON, REL, LCL, SHR, -
    0000                          EXE, RD, NOWRT, LONG
    0000
```

```
                 0000         .SBTTL  PAS$GOTO - Perform up-level GOTO
                 0000   ;++
                 0000   ; FUNCTIONAL DESCRIPTION:
                 0000   ;
                 0000   ;       This procedure is called by PASCAL compiled code to perform
                 0000   ;       an up-level GOTO.  Functionally, it performs a $UNWIND to
                 0000   ;       the specified frame and PC.  The actual implementation is
                 0000   ;       described in detail below.
                 0000   ;
                 0000   ; CALLING SEQUENCE:
                 0000   ;
                 0000   ;       CALL PAS$GOTO (dest_FP.ra.v, dest_PC.jzi.r)
                 0000   ;
                 0000   ; FORMAL PARAMETERS:
                 0000   ;
        00000004 0000           dest_FP = 4             ; The FP of the destination frame
                 0000                                   ; If the signal is PAS$_GOTO, it has two
                 0000                                   ; "FAO arguments", the destination FP and PC.
                 0000
        00000008 0000           dest_PC = 8             ; The PC of the destination instruction
                 0000
                 0000   ;
                 0000   ; IMPLICIT INPUTS:
                 0000   ;
                 0000   ;       NONE
                 0000   ;
                 0000   ; IMPLICIT OUTPUTS:
                 0000   ;
                 0000   ;       NONE
                 0000   ;
                 0000   ; COMPLETION STATUS:
                 0000   ;
                 0000   ;       NONE
                 0000   ;
                 0000   ; SIDE EFFECTS:
                 0000   ;
                 0000   ;       Functionally performs a $UNWIND to the specified FP and PC.
                 0000   ;
                 0000   ;--
```

```
0000   ;+
0000   ; Implementation notes:
0000   ;
0000   ;       An "up-level" GOTO is a GOTO where the destination is not in the
0000   ;       same stack frame (procedure incarnation) as the origination.  Ideally,
0000   ;       what one wants to do is unwind the stack frames back to the
0000   ;       desired frame, and then begin executing at the destination labelled
0000   ;       instruction.  The unwind is necessary to restore saved registers;
0000   ;       one can't simply JMP to the instruction since the stack frame and
0000   ;       register contents would be inconsistent.
0000   ;
0000   ;       There is, of course, the system service $UNWIND that seems to do
0000   ;       exactly what we want.  You specify to $UNWIND the number of frames
0000   ;       to remove and the desired PC, and off it goes.  The first problem
0000   ;       with this is that you can only unwind while in a condition handler
0000   ;       (or in a procedure called from a condition handler).  This is not
0000   ;       much of a problem, one can simply signal a special exception and
0000   ;       intercept it in a handler, which then does the $UNWIND.  The
0000   ;       second problem is that, while $UNWIND wants a number of frames to
0000   ;       remove, we don't know how many frames distant we are from the
0000   ;       destination; we do know the FP value of the destination frame.  So,
0000   ;       the initial implementation searched through the stack frame chain,
0000   ;       counting until it found the desired FP.  It then signalled PAS$_GOTO
0000   ;       with arguments of the count and PC, and its own handler did the
0000   ;       unwind.  This worked well in normal cases, but failed spectacularly
0000   ;       if it was called from a condition handler.
0000   ;
0000   ;       The problem was simply that when $UNWIND counts stack frames, and
0000   ;       it comes across a condition handler, it "skips" to the establisher's
0000   ;       frame without counting intervening frames.  This is correct according
0000   ;       to the handler search algorithm.  Because PAS$GOTO wasn't taking this
0000   ;       into account, the number of frames to unwind that it specified was
0000   ;       wrong.
0000   ;
0000   ;       It is difficult, though possible, to have PAS$GOTO count frames in the
0000   ;       same manner as $UNWIND.  If this is done, one finds another problem;
0000   ;       this time due to a design flaw in $UNWIND.  Basically, if one
0000   ;       specifies a non-zero number of frames to unwind, with the intention
0000   ;       of unwinding to an establisher's frame, $UNWIND removes one stack frame
0000   ;       too many.  If you decrease the count by one, you unwind only to
0000   ;       the handler.  Thus, it is impossible to unwind exactly to an establisher
0000   ;       frame if that signal is not the current one.  Since being able to
0000   ;       GOTO elsewhere in the establisher's frame is a desireable feature, this
0000   ;       is unacceptable.
0000   ;
0000   ;       An intermediate implementation was tried which simply establishes
0000   ;       a handler in the destination frame, signals PAS$_GOTO, and lets
0000   ;       that handler unwind to its establisher.  This doesn't work when there
0000   ;       is already a signal in progress since the special GOTO handler is
0000   ;       skipped.
0000   ;
0000   ;       The successful solution is somewhat complicated, and is actually
0000   ;       two solutions in one.  There are two interesting cases of an
0000   ;       up-level GOTO:
0000   ;               1. There is no signal currently in progress
0000   ;               2. There are one or more signals currently in progress
0000   ;
```

```
0000   ;       The first case can be solved with either the original method or
0000   ;       with the ''intermediate implementation'' where a special handler
0000   ;       is temporarily established in the destination frame.  The latter
0000   ;       is what is used; PAS$HANDLER, if already established, serves as
0000   ;       that special handler, or PAS$$UNWIND_GOTO is established if there
0000   ;       is no handler.  We assume that no handler other than PAS$HANDLER
0000   ;       is established in the destination frame.  This is reasonable, because
0000   ;       the only way it could get there is by the user calling LIB$ESTABLISH,
0000   ;       and if this was done, the user got a compile-time warning from
0000   ;       VAX-11 PASCAL saying that LIB$ESTABLISH was incompatible with
0000   ;       VAX-11 PASCAL.
0000   ;
0000   ;       Once a handler is established in the destination frame, PAS$_GOTO
0000   ;       is signalled, with two FAO parameters of the destination FP and
0000   ;       PC.  Note that the stack-frame search of the original implementation
0000   ;       is no longer present.  If no other exception is in progress, this
0000   ;       signal will be caught by the handler in the destination frame, which
0000   ;       will then unwind zero frames to the establisher at the destination PC.
0000   ;
0000   ;       The more interesting case is when there is an exception in progress.
0000   ;       PAS$HANDLER, which was established when the user used the ESTABLISH
0000   ;       builtin, and which has already been called for the current signal,
0000   ;       has itself established a handler PAS$$GOTO_HANDLER.  This handler
0000   ;       causes an unwind back to the frame of its establisher (PAS$HANDLER),
0000   ;       but at PC UNWIND_TO_ESTABLISHER.  This effectively removes the last
0000   ;       exception (PAS$_GOTO).  Before unwinding, the destination FP and PC
0000   ;       are loaded into the saved R0 and R1 so that they can be communicated
0000   ;       to UNWIND_TO_ESTABLISHER.
0000   ;
0000   ;       UNWIND_TO_ESTABLISHER then unwinds zero frames to the establisher, but
0000   ;       at PC JUMP_TO_DEST.  Again, R0 and R1 have the saved FP and PC.
0000   ;       JUMP_TO_DEST looks to see if the destination FP is the same as its
0000   ;       current FP, which it might not be.  If it is, then it simply jumps
0000   ;       to the destination PC.  Otherwise, it calls PAS$GOTO again with
0000   ;       the original arguments.  Eventually, all signals between the source
0000   ;       and the destination of the GOTO will be unwound.
0000   ;
0000   ;       The following problems with $UNWIND are known:
0000   ;            1.  You can't unwind more than one exception reliably.
0000   ;            2.  Unwinding zero frames leaves the signal and mechanism arglists,
0000   ;                along with some other stuff, on the stack.  This doesn't
0000   ;                bother us as PASCAL always readjusts the stack at GOTO
0000   ;                destinations.
0000   ;            3.  Unwinding zero frames doesn't restore the saved R0 and R1
0000   ;                from the mchargs list.  This is solved by manually loading
0000   ;                the registers before doing the RET from the handler.
0000   ;-
```

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

```
                            0020         .SBTTL  PAS$$GOTO_HANDLER - Established by PAS$HANDLER
                            0020  ;++
                            0020  ; FUNCTIONAL DESCRIPTION:
                            0020  ;
                            0020  ;      This is a condition handler established by PAS$HANDLER which
                            0020  ;      intercepts PAS$_GOTO exceptions and unwinds back to its
                            0020  ;      establisher's frame (PAS$HANDLER) but at PC UNWIND_TO_ESTABLISHER.
                            0020  ;
                            0020  ; CALLING SEQUENCE:
                            0020  ;
                            0020  ;      ret-status = PAS$$GOTO_HANDLER (sigargs.rlu.r, mchargs.rlu.r)
                            0020  ;
                            0020  ; FORMAL PARAMETERS:
                            0020  ;
                            0020
                  00000004  0020         sigargs = 4                ; The signal arguments list
                            0020                                    ; If the signal is PAS$_GOTO, it has two
                            0020                                    ; "FAO arguments", the destination FP and PC.
                            0020
                  00000008  0020         mchargs = 8                ; The mechanism arguments list
                            0020
                            0020  ;
                            0020  ; IMPLICIT INPUTS:
                            0020  ;
                            0020  ;      NONE
                            0020  ;
                            0020  ; IMPLICIT OUTPUTS:
                            0020  ;
                            0020  ;      NONE
                            0020  ;
                            0020  ; COMPLETION STATUS:
                            0020  ;
                            0020  ;      NONE
                            0020  ;
                            0020  ; SIDE EFFECTS:
                            0020  ;
                            0020  ;      Unwinds back to its establisher (PAS$HANDLER), but at PC
                            0020  ;      UNWIND_TO_ESTABLISHER.
                            0020  ;
                            0020  ;--
                            0020
                      0000  0020         .ENTRY  PAS$$GOTO_HANDLER, ^M<>
          51    04 AC   D0  0022         MOVL    sigargs(AP), R1                    ; Get signal arguments list
 00000000'8F    04 A1   D1  0026         CMPL    CHF$L_SIG_NAME(R1), #PAS$_GOTO     ; Is this PAS$_GOTO?
                   06   13  002E         BEQL    10$                                ; If so, keep going
          50  0918 8F   3C  0030         MOVZWL  #SS$_RESIGNAL, R0                  ; Resignal this exception
                        04  0035         RET                                        ; Return to CHF
                            0036
                            0036  ;+
                            0036  ; Unwind the stack frames back to our establisher, PAS$$HANDLER.  Use the
                            0036  ; saved R0 and R1 to communicate the destination FP and PC.
                            0036  ;-
                            0036
          50    08 AC   D0  0036  10$:   MOVL    mchargs(AP), R0                    ; Get mechanism arguments list
       0C A0    0C A1   7D  003A         MOVQ    12(R1), CHF$L_MCH_SAVR0(R0)        ; Store destination FP and PC
                            003F                                                    ; in saved R0 and R1
               61'AF    9F  003F         PUSHAB  B^UNWIND_TO_ESTABLISHER            ; Unwind to UNWIND_TO_ESTABLISHER
```

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

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

```
                            008E            .SBTTL  PAS$$UNWIND_GOTO - Unwind to destination FP and PC
                            008E    ;++
                            008E    ; FUNCTIONAL DESCRIPTION:
                            008E    ;
                            008E    ;       This is a condition handler established by PAS$GOTO in the
                            008E    ;       destination frame of an up-level GOTO.  It intercepts PAS$_GOTO
                            008E    ;       exceptions and initiates an unwind back to the destination
                            008E    ;       frame and PC.  This routine is also called by PAS$HANDLER if
                            008E    ;       it detects PAS$_GOTO.
                            008E    ;
                            008E    ; CALLING SEQUENCE:
                            008E    ;
                            008E    ;       ret-status = PAS$$UNWIND_GOTO (sigargs.rlu.r, mchargs.rlu.r)
                            008E    ;
                            008E    ; FORMAL PARAMETERS:
                            008E    ;
                            008E
                 00000004   008E            sigargs = 4                     ; The signal arguments list
                            008E                                            ; If the signal is PAS$_GOTO, it has two
                            008E                                            ; "FAO arguments", the destination FP and PC.
                            008E
                 00000008   008E            mchargs = 8                     ; The mechanism arguments list
                            008E
                            008E    ;
                            008E    ; IMPLICIT INPUTS:
                            008E    ;
                            008E            NONE
                            008E    ;
                            008E    ; IMPLICIT OUTPUTS:
                            008E    ;
                            008E            NONE
                            008E    ;
                            008E    ; COMPLETION STATUS:
                            008E    ;
                            008E            NONE
                            008E    ;
                            008E    ; SIDE EFFECTS:
                            008E    ;
                            008E            Unwinds back to the destination frame and PC.
                            008E    ;
                            008E    ;--
                            008E
                     0004   008E            .ENTRY  PAS$$UNWIND_GOTO, ^M<R2>
             50   04 AC  7D 0090            MOVQ    sigargs(AP), R0                 ; Get signal and mechanism lists
    00000000'8F   04 A0  D1 0094            CMPL    CHF$L_SIG_NAME(R0), #PAS$_GOTO  ; Is this PAS$_GOTO?
                     07  12 009C            BNEQ    10$                             ; If not, resignal
          04 A1   0C A0  D1 009E            CMPL    12(R0), CHF$L_MCH_FRAME(R1)     ; Is establisher FP the dest FP?
                     06  13 00A3            BEQL    20$                             ; Skip if so
          50    0918 8F  3C 00A5    10$:    MOVZWL  #SS$_RESIGNAL, R0               ; Resignal this exception
                         04 00AA            RET                                     ; Return to CHF
                            00AB
                            00AB    ;+
                            00AB    ; If the handler established in our "establisher's" frame is PAS$$UNWIND_GOTO,
                            00AB    ; (which it wouldn't be if we were called from PAS$HANDLER), remove our
                            00AB    ; address as that frame's handler.
                            00AB    ;-
                            00AB
          52    E0 AF    9E  00AB  20$:    MOVAB   B^PAS$$UNWIND_GOTO, R2           ; Get address of our entry mask
          04 B1    52    D1  00AF          CMPL    R2, @CHF$L_MCH_FRAME(R1)         ; Is it the same as establishers
                             00B3                                                   ; handler?
                   03    12  00B3          BNEQ    30$                              ; Skip if not
                04 B1    D4  00B5          CLRL    @CHF$L_MCH_FRAME(R1)             ; Cancel the handler
                             00B8
                             00B8  ;+
                             00B8  ; Unwind the stack frames back to our establisher, the destination frame,
                             00B8  ; and to the destination PC.
                             00B8  ;-
                             00B8
                10 A0    DD  00B8  30$:    PUSHL   16(R0)                           ; Push destination PC
                08 A1    9F  00BB          PUSHAB  CHF$L_MCH_DEPTH(R1)              ; Unwind to establisher
       00000000'GF  02   FB  00BE          CALLS   #2, G^SYS$UNWIND                 ; Do the unwind
                03 50    E8  00C5          BLBS    R0, 40$                          ; Return if successful
                 FF85    31  00C8          BRW     UNWIND_FAILED                    ; Signal failure of UNWIND
                         04  00CB  40$:    RET                                      ; Return to UNWIND service
                             00CC
                             00CC          .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)
```

0294 AH-BT13A-SE
VAX/VMS V4.0

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