```
PPPPPPPPPPPP   LLL             IIIIIIIII   RRRRRRRRRRRR   TTTTTTTTTTTTTTT   LLL
PPPPPPPPPPPP   LLL             IIIIIIIII   RRRRRRRRRRRR   TTTTTTTTTTTTTTT   LLL
PPPPPPPPPPPP   LLL             IIIIIIIII   RRRRRRRRRRRR   TTTTTTTTTTTTTTT   LLL
PPP      PPP   LLL                III      RRR      RRR         TTT         LLL
PPP      PPP   LLL                III      RRR      RRR         TTT         LLL
PPP      PPP   LLL                III      RRR      RRR         TTT         LLL
PPP      PPP   LLL                III      RRR      RRR         TTT         LLL
PPP      PPP   LLL                III      RRR      RRR         TTT         LLL
PPP      PPP   LLL                III      RRR      RRR         TTT         LLL
PPPPPPPPPPPP   LLL                III      RRRRRRRRRRRR         TTT         LLL
PPPPPPPPPPPP   LLL                III      RRRRRRRRRRRR         TTT         LLL
PPPPPPPPPPPP   LLL                III      RRRRRRRRRRRR         TTT         LLL
PPP            LLL                III      RRR   RRR            TTT         LLL
PPP            LLL                III      RRR   RRR            TTT         LLL
PPP            LLL                III      RRR   RRR            TTT         LLL
PPP            LLL                III      RRR      RRR         TTT         LLL
PPP            LLL                III      RRR      RRR         TTT         LLL
PPP            LLL                III      RRR      RRR         TTT         LLL
PPP            LLLLLLLLLLLLLLL IIIIIIIII   RRR         RRR      TTT         LLLLLLLLLLLLLLL
PPP            LLLLLLLLLLLLLLL IIIIIIIII   RRR         RRR      TTT         LLLLLLLLLLLLLLL
PPP            LLLLLLLLLLLLLLL IIIIIIIII   RRR         RRR      TTT         LLLLLLLLLLLLLLL
```

FILEID**PLICONDIT


```
PPPPPPPP    LL            IIIIII        CCCCCCCC    OOOOOO    NN      NN  DDDDDDDD      IIIIII    TTTTTTTTTT
PPPPPPPP    LL            IIIIII        CCCCCCCC    OOOOOO    NN      NN  DDDDDDDD      IIIIII    TTTTTTTTTT
PP      PP  LL              II      CC            OO      OO  NN      NN  DD      DD      II          TT
PP      PP  LL              II      CC            OO      OO  NN      NN  DD      DD      II          TT
PP      PP  LL              II      CC            OO      OO  NNNN    NN  DD      DD      II          TT
PP      PP  LL              II      CC            OO      OO  NNNN    NN  DD      DD      II          TT
PPPPPPPP    LL              II      CC            OO      OO  NN  NN  NN  DD      DD      II          TT
PPPPPPPP    LL              II      CC            OO      OO  NN  NN  NN  DD      DD      II          TT
PP          LL              II      CC            OO      OO  NN    NNNN  DD      DD      II          TT
PP          LL              II      CC            OO      OO  NN    NNNN  DD      DD      II          TT
PP          LL              II      CC            OO      OO  NN      NN  DD      DD      II          TT      ....
PP          LL              II      CC            OO      OO  NN      NN  DD      DD      II          TT      ....
PP          LLLLLLLLLL    IIIIII        CCCCCCCC    OOOOOO    NN      NN  DDDDDDDD      IIIIII        TT      ....
PP          LLLLLLLLLL    IIIIII        CCCCCCCC    OOOOOO    NN      NN  DDDDDDDD      IIIIII        TT      ....

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

PL
1-

```
0000        .title  pli$condit -  condition handler routines
0000        .ident  /1-008/                                       ; Edit DSB1008
0000                                                              ; Edit DSB1007
0000                                                              ; Edit CGN1006
0000                                                              ; Edit CGN1005
0000                                                              ; Edit CGN1004
0000                                                              ; Edit CGN1003
0000                                                              ; Edit WHM1002
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  ; facility:
0000  ;
0000  ;       VAX/VMS PL1 runtime library.
0000  ;
0000  ; abstract:
0000  ;
0000  ;       This module contains the pl1 runtime routines for condition
0000  ;       handling and non-local goto processing.
0000  ;
0000  ; author: r. heinen 18-jan-1979
0000  ;
0000  ; Modifications:
0000  ;
0000  ;
0000  ;       1-002   Bill Matthews   29-September-1982
0000  ;
0000  ;               Invoke macros $defdat and rtshare instead of $defopr and share.
0000  ;
0000  ;       1-003   Chip Nylander   25-January-1983
0000  ;
0000  ;               Fix the ONCODE() bif.  It was blown to bits by a previous
0000  ;               edit, and did not return the correct results for most cases.
0000  ;               The new, and correct, behavior is:
```

```
0000        ;
0000        ;               1) For PLI$_ENDFILE, PLI$_ENDPAGE, PLI$_FINISH, and all
0000        ;                  non-PLI conditions, return the primary condition code.
0000        ;                  Note that FIXEDOVERFLOW, OVERFLOW, UNDERFLOW, and
0000        ;                  ZERODIVIDE are not signalled as PL/I conditions.
0000        ;
0000        ;               2) for all other conditions (currently ERROR, KEY,
0000        ;                  UNDEFINEDFILE, and VAXCONDITION) search the signal
0000        ;                  argument list and return the last (most specific)
0000        ;                  condition, but ignore all instances of PLI$_FILENAME,
0000        ;                  PLI$_RMSF, PLI$_RMSR, and PLI$_IOERROR because they
0000        ;                  are "uninteresting".
0000        ;
0000        ;       1-004   Chip Nylander   07-February-1983
0000        ;
0000        ;               Make floating faults visible to ON units.
0000        ;
0000        ;       1-005   Chip Nylander   22-February-1983
0000        ;
0000        ;               Prevent spurious error messages when a VAX condition is
0000        ;               signalled and not handled:
0000        ;               when calling LIB$SIGNAL from the OPTIONS(MAIN) condition
0000        ;               to make PLI$_ERROR the primary error, don't pass the saved
0000        ;               psl/pc to LIB$SIGNAL for those system errors that don't
0000        ;               want it.
0000        ;
0000        ;       1-006   Chip Nylander   17-March-1983
0000        ;
0000        ;               Make SIGNAL_ERROR a local symbol rather than a global one.
0000        ;
0000        ;       1-007   Dave Blickstein 14-February-1984
0000        ;
0000        ;               Fixup norm_exit in PL$DEF_HND to resignal to the system
0000        ;               handler when PLIG-ERROR is received with I severity.
0000        ;               It was signalling FINISH before passing the error on.
0000        ;
0000        ;       1-008   Dave Blickstein 23-May-1984
0000        ;
0000        ;               Changed PLI$OPTMAIN_HND to resignal SS$_DEBUG.  This was
0000        ;               done so that PL/I programs can be ^Y-ed and examined with
0000        ;               the debugger with the DEBUG command.  Previously, SS$_DEBUG
0000        ;               would be resignalled like any other error and a traceback
0000        ;               would occur followed by an image exit.
0000        ;--
0000
0000        ;
0000        ; external definitions
0000        ;
0000
0000                $chfdef                         ; define condition handler offsets
0000                $defcnd                         ; define condition handler block
0000                $deffcb                         ; define runtime file control block
0000                $defstk                         ; define runtime stack
0000                $fabdef                         ; define fab offsets
0000                $rabdef                         ; define rab offsets
0000                $namdef                         ; define nam offsets
0000                $sfdef                          ; define stack offset
```

```
                                                0000
                                                0000   ;
                                                0000   ; local definitions
                                                0000   ;
                                     00000000   0000           zerodiv = 0
                                     00000004   0000           anycond = 4
                                     00000008   0000           intovf  = 8
                                                0000
                                                0000
                                                0000           rtshare
                                                0000   ;
                                                0000   ; table of subscript range message numbers
                                                0000   ;
                                                0000   subscript_table:
                                    00000000'   0000           .long   pli$_subrange
                                    00000000'   0004           .long   pli$_subrange1
                                    00000000'   0008           .long   pli$_subrange2
                                    00000000'   000C           .long   pli$_subrange3
                                    00000000'   0010           .long   pli$_subrange4
                                    00000000'   0014           .long   pli$_subrange5
                                    00000000'   0018           .long   pli$_subrange6
                                    00000000'   001C           .long   pli$_subrange7
                                    00000000'   0020           .long   pli$_subrange8
00000000 00000000 00000000 00000000   0024           .long   0,0,0,0,0,0,0,0,0,0,0
00000000 00000000 00000000 00000000   0034
         00000000 00000000 00000000   0044
                                    00000000'   0050           .long   pli$_substr2
                                    00000000'   0054           .long   pli$_substr3
                                                0058
                                                0058   ;
                                                0058   ; table of hardware-raised errors that need the PSL and PC returned to DCL
                                                0058   ; for FAO formatting.
                                                0058   ;
                                                0058   hardware_table:
                                    00000000'   0058           .long   ss$_intovf
                                    00000000'   005C           .long   ss$_intdiv
                                    00000000'   0060           .long   ss$_fltovf
                                    00000000'   0064           .long   ss$_fltovf_f
                                    00000000'   0068           .long   ss$_fltund
                                    00000000'   006C           .long   ss$_fltund_f
                                    00000000'   0070           .long   ss$_fltdiv
                                    00000000'   0074           .long   ss$_fltdiv_f
                                    00000000'   0078           .long   ss$_decovf
                                    00000000'   007C           .long   ss$_subrng
                                    00000000'   0080           .long   ss$_accvio
                                    00000000'   0084           .long   ss$_roprand
                                    00000000'   0088           .long   ss$_radrmod
                                    00000000'   008C           .long   ss$_opcdec
                                    00000000'   0090           .long   ss$_opccus
                                     00000000   0094           .long   0                       ; must be last
                                                0098
```

```
                              0098       .sbttl  pli$def_hnd - pl1 default condition handler
                              0098  ;++
                              0098  ; pli$def_hnd - pl1 default condition handler
                              0098  ;
                              0098  ; functional description:
                              0098  ;
                              0098  ; This routine is the condition handler for all pl1 main programs.
                              0098  ; The action is to search the normal on unit list in the same way
                              0098  ; as the standard handler. If a condition handler was found then the
                              0098  ; condition is continued. If no condition handler was found then error is signaled.
                              0098  ;
                              0098  ; inputs:
                              0098  ;
                              0098  ;       condition argument list
                              0098  ;
                              0098  ; outputs:
                              0098  ;
                              0098  ;       none, but the program may be aborted.
                              0098  ; Two entry points are provided so that the remainder of the runtime
                              0098  ; can distinguish the highest level options main handler
                              0098  ;--
                        0FFC  0098       .entry  pli$optmain_hnd,^m<r2,r3,r4,r5,r5,r6,r7,r8,r9,r10,r11>
                   02     11  009A       brb     def_hnd_common          ; use common code for handler
                        0FFC  009C       .entry  pli$def_hnd,^m<r2,r3,r4,r5,r6,r7,r8,r9,r10,r11>;
                              009E  def_hnd_common:
                              009E  ;
                              009E  ; search this frames condition list for an on unit
                              009E  ;
                 012A     30  009E       bsbw    search_cnd_list         ; look for a handler in this frame
          52    04 AC     D0  00A1       movl    chf$l_sigarglst(ap),r2  ; address the signal arg list
                              00A5  ;
                              00A5  ; if an on unit was found then continue
                              00A5  ;
     00000000'8F      50  D1  00A5       cmpl    r0,#ss$_continue        ; handler found?
                      01  12  00AC       bneq    10$                     ; if neq then perform default action
                          04  00AE       ret                             ; continue after user handler
                              00AF
                              00AF  ;
                              00AF  ; perform default action based on the condition value
                              00AF  ;
                              00AF  10$:
       0000'8F    06 A2   B1  00AF       cmpw    chf$l_sig_name+2(r2),#<pli$_error@-16>; pl1 defined?
                      1F  12  00B5       bneq    20$                     ; signal pl1 error condition
  50    04 A2   0A    03  EF  00B7       extzv   #3,#10,chf$l_sig_name(r2),r0; get message number
          0000'8F     50  B1  00BD       cmpw    r0,#<pli$_finish&^xfff>@-3; finish?
                      3F  13  00C2       beql    23$                     ; if eql then continue
                              00C4       case    r0,<-
                              00C4               signal_error,-          ; never signaled
                              00C4               end_of_page,-           ; end page
                              00C4               signal_error,-          ; end file
                              00C4               signal_error,-          ; unknown file
                              00C4               signal_error,-          ; key
                              00C4               norm_exit-              ; error signaled, so exit
                              00C4               >
                      57  11  00D4       brb     signal_error            ; all others cause error signal
                              00D6  ;
                              00D6  ; non pl1 condition
```

```
                                    00D6      ;
         00000000'8F    04 A2   D1  00D6      20$:    cmpl    chf$l_sig_name(r2),#ss$_fltund; floating underflow trap?
                           29   13  00DE              beql    25$                   ; if eql then ignore
         00000000'8F    04 A2   D1  00E0              cmpl    chf$l_sig_name(r2),#ss$_fltund_f; floating underflow fault?
                           1F   13  00E8              beql    25$                   ; if eql then ignore
         00000000'8F    04 A2   D1  00EA              cmpl    chf$l_sig_name(r2),#ss$_debug   ; if activate debugger
                           03   12  00F2              bneq    22$                   ;
                         00C9   31  00F4              brw     finish_signal_return  ; then resignal
00000000'8F    04 A2    1D  03  ED  00F7      22$:    cmpzv   #3,#29,chf$l_sig_name(r2),#<ss$_unwind@-3>; unwind?
                           2A   12  0101              bneq    signal_error          ; all others get default handling
                    00000000'EF 16  0103      23$:    jsb     pli$$term_prog        ; shut off program exit control
               50     0000'8F   3C  0109      25$:    movzwl  #ss$_continue,r0      ; continue
                           04       010E              ret
                                    010F      ;
                                    010F      ; end of page
                                    010F      ;
                                    010F      end_of_page:
                   52    08 AC  D0  010F              movl    chf$l_mcharglst(ap),r2  ;get mech args
                   5C    0C A2  D0  0113              movl    chf$l_mch_savr0(r2),ap  ;set fcb addr
                           50   D4  0117              clrl    r0                      ;set no line
                   51      01   D0  0119              movl    #1,r1                   ;set to put page
                   52      00   7D  011C              movq    #0,r2                   ;set no skip, no options
                           54   D4  011F              clrl    r4                      ;set no format
                    00000000'GF 16  0121              jsb     g^pli$putfile_r6        ;do put page
                    00000000'GF 17  0127              jmp     g^pli$put_end_r6        ;finish it off and return
                                    012D      ;
                                    012D      ; signal error condition
                                    012D      ;
                                    012D      signal_error:
                        10 AD   DD  012D              pushl   stk_l_pc(fp)            ;save system pc/psl
                        04 AD   DD  0130              pushl   stk_l_psl(fp)           ;
                   50      62   D0  0133              movl    chf$l_sig_args(r2),r0   ;get size of arg list
            10 AD    FC A240    D0  0136              movl    chf$l_sig_args-4(r2)[r0],stk_l_pc(fp) ;set original pc/psl
            04 AD      6240     D0  013C              movl    chf$l_sig_args(r2)[r0],stk_l_psl(fp) ;
               53    FF13 CF    DE  0141              moval   hardware_table,r3       ; set to top of hardware error table
                           63   D5  0146      10$:    tstl    (r3)                    ; any more?
                           0C   13  0148              beql    30$                     ; no, so throw away pc/psl
               83    04 A2      D1  014A              cmpl    chf$l_sig_name(r2),(r3)+; need pc/psl?
                           F6   12  014E              bneq    10$                     ; if not, loop back to check again
               53    50    02   C1  0150              addl3   #2,r0,r3                ;get number of args for signal
                           06   11  0154              brb     50$                     ; rejoin common
                   53      50   D0  0156      30$:    movl    r0,r3                   ;copy number of args for signal
                   50      02   C2  0159              subl    #2,r0                   ;set number of args to copy less pc/psl
                         6240   DD  015C      50$:    pushl   chf$l_sig_args(r2)[r0]  ;copy args to stack
                      FA   50   F5  015F              sobgtr  r0,50$                  ;
                           00   DD  0162              pushl   #0                      ;set no args for error
                  00000000'8F   DD  0164              pushl   #pli$_error             ;set error
         50    04 A2   03  00   EF  016A              extzv   #0,#3,chf$l_sig_name(r2),r0; get severity of previous condition
            6E    03   00  50   F0  0170              insv    r0,#0,#3,(sp)           ; insert as severity of this signal
                           6D   D4  0175              clrl    (fp)                    ;disappear our condition handler
                           50   7C  0177              clrq    r0                      ;set no secondary arguments
            00000000'GF    53   FB  0179              calls   r3,g^lib$signal         ;signal error
                                    0180      error_signal:                           ;if we come back, the signal was handled
                      04 AD  8ED0   0180              popl    stk_l_psl(fp)           ;restore system pc/psl
                      10 AD  8ED0   0184              popl    stk_l_pc(fp)            ;
               50     00'8F     9A  0188              movzbl  #ss$_continue,r0        ;set to continue
                           04       018C              ret                             ;return
```

```
                                 018D    ;
                                 018D    ; end the program
                                 018D    ;
                                 018D    norm_exit:
04   04 A2   03   00     ED      018D            cmpzv   #0,#3,chf$l_sig_name(r2),#4; fatal severity?
                  2B     19      0193            blss    finish_signal_return    ; if leq then no
          00000000'GF    16      0195            jsb     g^pli$$term_prog        ; shut off exit handler finish condition
                10 AD    DD      019B            pushl   stk_l_pc(fp)            ;save system pc/psl
                04 AD    DD      019E            pushl   stk_l_psl(fp)           ;
      10 AD    B8'AF     9E      01A1            movab   b^10$,stk_l_pc(fp)      ;set unrecognized pc/psl
                04 AD    DC      01A6            movpsl  stk_l_psl(fp)           ;
          00000000'8F    DD      01A9            pushl   #pli$_finish            ; signal finish condition
                   6D    D4      01AF            clrl    (fp)                    ; remove trouble handler
     00000000'GF   01    FB      01B1            calls   #1,g^lib$signal
             04 AD     8ED0      01B8    10$:    popl    stk_l_psl(fp)           ;restore system pc/psl
             10 AD     8ED0      01BC            popl    stk_l_pc(fp)            ;
                                 01C0    ;
                                 01C0    ; print messages via system or next handler
                                 01C0    ;
                                 01C0    finish_signal_return:
      50    0000'8F      3C      01C0            movzwl  #ss$_resignal,r0        ; get the messages
                         04      01C5            ret                             ; just return
```

```
             01C6        .sbttl  pli$cnd_hnd - condition handler for pl1
             01C6  ;++
             01C6  ; pli$cnd_hnd - condition handler for pl1
             01C6  ;
             01C6  ; functional description:
             01C6  ;
             01C6  ; This routine handles conditions for pl1 programs by searching the
             01C6  ; current condition handler list for a control block which specifies
             01C6  ; the same enable. The first control block with a matching enable is
             01C6  ; used to call the associated condition handler routine.
             01C6  ; If no handler is found then the condition is resignaled.
             01C6  ;
             01C6  ; inputs:
             01C6  ;
             01C6  ;         exception stack frame
             01C6  ;         The signaller's r1 is meaningfull on known pl1 conditions.
             01C6  ;         If the condition relates to a file then that r1 is the fcb address
             01C6  ;         of the file.
             01C6  ;
             01C6  ; outputs:
             01C6  ;
             01C6  ;         none
             01C6  ;
             01C6  ; If an error occurs while handling the exception, a fatal exit is taken.
             01C6  ; A signal will not work because control will return here.
             01C6  ;
             01C6  ; The call to the condition handler is done with the argument list of the condition.
             01C6  ;
             01C6  ;--
             01C6        .enabl  lsb
      001C   01C6        .entry  pli$cnd_hnd,^m<r2,r3,r4>
   01   10   01C8        bsbb    search_cnd_list          ; search the condition list
        04   01CA        ret                              ; done
```

```
                                  01CB          .sbttl search_cnd_list
                                  01CB  ;
                                  01CB  ; subroutine to search the establisher's condition handler list
                                  01CB  ;
                                  01CB  search_cnd_list:
          6D   0321'CF    9E      01CB          movab   w^fatal_exception,(fp)  ; set up fatal exception vector
                    7E    7C      01D0          clrq    -(sp)                   ; zero stack locals
                    7E    D4      01D2          clrl    -(sp)                   ;
          51     04 AC    D0      01D4          movl    chf$l_sigarglst(ap),r1  ; get the signal value
          52     04 A1    D0      01D8          movl    chf$l_sig_name(r1),r2   ;
             54     61    D0      01DC          movl    chf$l_sig_args(r1),r4   ; get argument numbers
          54   FC A144    DE      01DF          moval   chf$l_sig_args-4(r1)[r4],r4; address the pc
          53     08 AC    D0      01E4          movl    chf$l_mcharglst(ap),r3  ; address arg list with fp of establisher
          50     0C A3    D0      01E8          movl    chf$l_mch_savr0(r3),r0  ;get fcb addr if its there
          51     04 A3    D0      01EC  1$:     movl    chf$l_mch_frame(r3),r1  ;
                                  01F0  ;
                                  01F0  ; search condition list in that frame
                                  01F0  ;
          53     F4 A1    9E      01F0          movab   stk_l_cnd_lst(r1),r3    ; address condition handler control list
             53     63    D0      01F4  5$:     movl    (r3),r3                 ; get next block
                    03    12      01F7          bneq    10$                     ; if eql then done
                  00DB    31      01F9          brw     40$                     ; resignal condition
                 08 A3    D5      01FC  10$:    tstl    cnd_l_addr(r3)          ; reverted condition handler?
                    F3    13      01FF          beql    5$                      ; if eql then yes
                                  0201  ;
                                  0201  ; look for fixed overflow handler
                                  0201  ;
00000001'8F    04 A3    1D    03  ED  0201      cmpzv   #3,#29,cnd_l_enabl(r3),#<pli$_zerodiv+8>@-3; PLI fixed overflow?
                    09    12      020B          bneq    15$                     ; if neq then no
                 08 AE    D5      020D          tstl    intovf(sp)              ; handler found already?
                    04    12      0210          bneq    15$                     ; if neq then yes
          08 AE     53    D0      0212          movl    r3,intovf(sp)           ; save for default case
                                  0216  ;
                                  0216  ; look for an anycondition handler
                                  0216  ;
00000000'8F    04 A3    1D    03  ED  0216  15$: cmpzv   #3,#29,cnd_l_enabl(r3),#<pli$_anycond>@-3; PLI any condition?
                    0B    12      0220          bneq    20$                     ; if neq then no
                 04 AE    D5      0222          tstl    anycond(sp)             ; already found in list?
                    06    12      0225          bneq    20$                     ; if neq then yes
          04 AE     53    D0      0227          movl    r3,anycond(sp)          ; save for default condition
                    13    11      022B          brb     25$                     ; continue
                                  022D  ;
                                  022D  ; look for zero divide handler
                                  022D  ;
00000000'8F    04 A3    1D    03  ED  022D  20$: cmpzv   #3,#29,cnd_l_enabl(r3),#<pli$_zerodiv>@-3; PLI zero divide
                    07    12      0237          bneq    25$                     ; if neq then no
                    6E    D5      0239          tstl    zerodiv(sp)             ; this condition enabled already?
                    03    12      023B          bneq    25$                     ; if neq then yes
             6E     53    D0      023D          movl    r3,zerodiv(sp)          ; save special dual value case
                                  0240  ;
                                  0240  ; look at secondary argument if PLI condition
                                  0240  ;
   00000000'8F    52    10    10  ED  0240  25$: cmpzv   #16,#16,r2,#<<pli$_error>@-16>; pl1 defined condition signaled?
                    42    13      0249          beql    29$                     ; if eql then yes
          52     04 A3    D1      024B          cmpl    cnd_l_enabl(r3),r2      ; enabled?
                    5B    13      024F          beql    30$                     ; if eql then signal condition
     00000000'8F 04 A3    D1      0251          cmpl    cnd_l_enabl(r3),#ss$_fltovf; enabled for overflow?
```

```
                   09   12  0259         bneq    26$                           ; no - check next fault
        00000000'8F  52   D1  025B         cmpl    r2,#ss$_fltovf_f              ; yes - is this an overflow fault?
                   48   13  0262         beql    30$                           ; if yes signal condition
       00000000'8F  04 A3  D1  0264  26$:  cmpl    cnd_l_enabl(r3),#ss$_fltund; enabled for underflow?
                   09   12  026C         bneq    27$                           ; no - check next fault
        00000000'8F  52   D1  026E         cmpl    r2,#ss$_fltund_f              ; yes - is this an underflow fault?
                   35   13  0275         beql    30$                           ; if yes signal condition
       00000000'8F  04 A3  D1  0277  27$:  cmpl    cnd_l_enabl(r3),#ss$_fltdiv; enabled for divide?
                   09   12  027F         bneq    28$                           ; no - go find next handler
        00000000'8F  52   D1  0281         cmpl    r2,#ss$_fltdiv_f              ; yes - is this a divide fault?
                   22   13  0288         beql    30$                           ; if yes signal condition
                 FF67   31  028A  28$:  brw     5$                            ; look for next handler
          7E  52   1D   03   EF  028D  29$:  extzv   #3,#29,r2,-(sp)               ; get internal bits
      8E  04 A3    1D   03   ED  0292         cmpzv   #3,#29,cnd_l_enabl(r3),(sp)+; match in pl1 bits?
                        F0   12  0298         bneq    28$                           ; if neq then no
00000000'8F  04 A3  1D  03   ED  029A         cmpzv   #3,#29,cnd_l_enabl(r3),#<pli$_key@-3>; file type?
                        06   1A  02A4         bgtru   30$                           ; if gtru then no additional arg
             50   0C A3  D1  02A6         cmpl    cnd_l_arg(r3),r0              ; same secondary argument?
                   DE   12  02AA         bneq    28$                           ; if neq then no
                            02AC  ;
                            02AC  ; call with r1 as establisher's frame pointer
                            02AC  ;
                   6D   D4  02AC  30$:  clrl    (fp)                          ; disable fatal exception vector
                04 AD   DD  02AE         pushl   stk_l_psl(fp)                 ; save system's psl
                10 AD   DD  02B1         pushl   stk_l_pc(fp)                  ; save system's stack key
           10 AD   64   D0  02B4         movl    (r4),stk_l_pc(fp)             ; set pc/psw to original
      04 AD   04 A4     B0  02B8         movw    4(r4),stk_l_psl(fp)           ; change psw only
           08 B3   6C   FA  02BD         callg   (ap),@cnd_l_addr(r3)          ; call handler with our arg list
                            02C1  norm_signal:                                  ; condition handler stack key
        6D   0321'CF   9E  02C1         movab   w^fatal_exception,(fp)        ; setup a condition handler
             10 AD   8ED0  02C6         popl    stk_l_pc(fp)                  ; restore system's stack key
             04 AD   8ED0  02CA         popl    stk_l_psl(fp)                 ; restore system's psl
        50   0000'8F   3C  02CE  35$:  movzwl  #ss$_continue,r0              ; set up to continue
            5E   0C    C0  02D3         addl    #12,sp                        ; purge stack
                  05       02D6         rsb                                   ; and do so
                            02D7  ;
                            02D7  ; no condition handler found - check defaults
                            02D7  ;
             53   6E   D0  02D7  40$:  movl    zerodiv(sp),r3                ; get zero divide handler
                   1B   13  02DA         beql    45$                           ; if eql then none
        00000000'8F  52   D1  02DC         cmpl    r2,#ss$_fltdiv                ; floating divide trap?
                   39   13  02E3         beql    60$                           ; if yes then merge
        00000000'8F  52   D1  02E5         cmpl    r2,#ss$_fltdiv_f              ; floating divide fault?
                   30   13  02EC         beql    60$                           ; if yes then merge
        00000000'8F  52   D1  02EE         cmpl    r2,#ss$_intdiv                ; integer divide?
                   27   13  02F5         beql    60$                           ; if eql then merge
          53   08 AE   D0  02F7  45$:  movl    intovf(sp),r3                 ; integer overflow?
                   12   13  02FB         beql    50$                           ; if eql then no handler found
        00000000'8F  52   D1  02FD         cmpl    r2,#ss$_decovf                ; decimal overflow?
                   18   13  0304         beql    60$                           ; if eql then handle condition
        00000000'8F  52   D1  0306         cmpl    r2,#ss$_intovf                ; integer overflow?
                   0F   13  030D         beql    60$                           ; if eql then handle
          53   04 AE   D0  030F  50$:  movl    anycond(sp),r3                ; any condition handler?
                   09   12  0313         bneq    60$                           ; if neq then handle
        50   0000'8F   3C  0315  55$:  movzwl  #ss$_resignal,r0              ; set up to continue signal
            5E   0C    C0  031A         addl    #12,sp                        ; purge stack
                  05       031D         rsb                                   ;
```

```
FF8B   31  031E   450 60$:     brw     30$                      ; signal contdition
           0321   451 ;
           0321   452 ; exception handler for exception handler
           0321   453 ;
           0321   454 fatal_exception:
     0000  0321   455          .word   0                        ;
           0323   456 fatal_error:
           0323   457          $exit_s #pli$_error              ; fatal error
           0330   458
           0330   459          .dsabl  lsb
```

```
                        0330          .sbttl  pli$nonloc_ret - non-local return processing
                        0330  ;++
                        0330  ; pli$nonloc_ret - non-local return
                        0330  ;
                        0330  ; functional description:
                        0330  ;
                        0330  ; This routine is entered via a jmp from a pl1 program when a non-local
                        0330  ; return occurs. The action is to unwind the call stack.
                        0330  ; The unwind is done in a manner similar to that of SYS$UNWIND.
                        0330  ;
                        0330  ; At each stack frame that is passed, the SS$_UNWIND condition is signaled.
                        0330  ;
                        0330  ; No provision is made for protecting r0/r1 from restoration by a call mask.
                        0330  ; PL/I does not save r0/r1
                        0330  ;
                        0330  ; inputs:
                        0330  ;
                        0330  ;         r0/r1 return value if any
                        0330  ;         r2 = number of frames to skip
                        0330  ;
                        0330  ; outputs:
                        0330  ;
                        0330  ;         none
                        0330  ;--
                        0330  pli$nonloc_ret::                                  ; not a call entry
      7E     50     7D  0330          movq    r0,-(sp)                   ; save return value and regs
      50     5D     D0  0333          movl    fp,r0                      ; copy current fp
                        0336  ;
                        0336  ; check for proper frame
                        0336  ;
10 A0     A7'AF     9E  0336  10$:    movab   b^canned_return,stk_l_pc(r0); force return in this module
   50     0C A0     D0  033B          movl    stk_l_fp(r0),r0            ; link to next frame on stack
 60   10     03     0D  033F          probew  #3,#stk_l_pc,(r0)          ; frame accessible?
             DE     13  0343          beql    fatal_error                ; if eql then insuff frames
          EE 52     F5  0345          sobgtr  r2,10$                     ; continue until done
                        0348
                        0348  ;
                        0348  ; proper frame found
                        0348  ;
      50     8E     7D  0348          movq    (sp)+,r0                   ; restore return value
             04         034B          ret                                ; unwind stack and goto
```

```
                        034C     ;++
                        034C     ; pli$optmain_ret - return in options main block
                        034C     ;
                        034C     ; functional description:
                        034C     ;
                        034C     ; This routine performs a return inside an options main block
                        034C     ; The action is to return unless this is the outermost options main block.
                        034C     ; If it the outermost block then a STOP is done.
                        034C     ;
                        034C     ; inputs:
                        034C     ;
                        034C     ;       none
                        034C     ;
                        034C     ; outputs:
                        034C     ;
                        034C     ;       none - r0,r1 are preserved for function returns
                        034C     ;--
                        034C     pli$optmain_ret::
                        034C     ;
                        034C     ; if this frame is the options main frame then STOP otherwise just return
                        034C     ;
        FD48 CF    9F   034C             pushab  w^pli$optmain_hnd         ; address options main handler
          6D 8E    D1   0350             cmpl    (sp)+,stk_l_cnd_hnd(fp)   ; this frame the outermost?
             09    12   0353             bneq    5$                        ; if neq then return
                        0355     ;
                        0355     ; cause a stop to occur
                        0355     ;
             50    DD   0355             pushl   r0                        ; set return value for exit code
 00000000'GF 01    FB   0357             calls   #1,g^pli$stop_prog        ; execute STOP
                   04   035E     5$:     ret                               ; continue
```

```
                         035F          .sbttl  pli$nonloc_goto - non-local goto processing
                         035F  ;++
                         035F  ; pli$nonloc_goto - non-local goto
                         035F  ;
                         035F  ; functional description:
                         035F  ;
                         035F  ; This routine is entered via a jmp from a pl1 program when a non-local
                         035F  ; goto occurs. The action is to unwind the call stack.
                         035F  ; The unwind is done in a manner similar to that of SYS$UNWIND.
                         035F  ;
                         035F  ; At each stack frame that is passed, the SS$_UNWIND condition is signaled.
                         035F  ;
                         035F  ; inputs:
                         035F  ;
                         035F  ;       r0 = new pc
                         035F  ;       r1 = fp of target frame
                         035F  ;
                         035F  ; outputs:
                         035F  ;
                         035F  ;       none
                         035F  ;--
                         035F  pli$nonloc_goto::
                         035F  pli$goto::                                       ; not a call entry
                         035F          .enabl  lsb
         5D     51   D1  035F          cmpl    r1,fp                            ; is this the frame?
                02   12  0362          bneq    10$                              ; if neq then no
                60   17  0364          jmp     (r0)                             ; goto new pc
                         0366  ;
                         0366  ; this goto is outside this stack frame
                         0366  ;
                50   DD  0366  10$:    pushl   r0                               ; save new pc
         50     5D   D0  0368          movl    fp,r0                            ; copy current fp
                         036B  ;
                         036B  ; check for proper frame
                         036B  ;
      0C A0     51   D1  036B  20$:    cmpl    r1,stk_l_fp(r0)                  ; is the next frame the last?
                2E   13  036F          beql    40$                              ; if eql then yes
     7E    FF4C CF   9E  0371          movab   w^norm_signal,-(sp)              ;address normal signal ret addr
     8E      10 A0   D1  0376          cmpl    stk_l_pc(r0),(sp)+               ;does this frame point there?
                12   13  037A          beql    25$                              ;if eql, yes, special case
     7E    FE00 CF   9E  037C          movab   w^error_signal,-(sp)             ;address error signal ret addr
     8E      10 A0   D1  0381          cmpl    stk_l_pc(r0),(sp)+               ;does this frame point there?
                07   13  0385          beql    25$                              ;if eql, yes, special case
   10 A0      A7'AF  9E  0387          movab   b^canned_return,stk_l_pc(r0); force return in this module
                05   11  038C          brb     27$                              ;cont
   10 A0      A6'AF  9E  038E  25$:    movab   b^unwind_signal,stk_l_pc(r0); force return for signal frame
     50       0C A0  D0  0393  27$:    movl    stk_l_fp(r0),r0                  ; link to next frame on stack
   60     10     03  0D  0397          probew  #3,#stk_l_pc,(r0)                ; frame accessible?
                86   13  039B          beql    fatal_error                      ; if eql then insuff frames
                CC   11  039D          brb     20$                              ; continue search
                         039F  ;
                         039F  ; proper frame found
                         039F  ;
        10 A0 8ED0       039F  40$:    popl    stk_l_pc(r0)                     ; force new pc on return
                50   7C  03A3          clrq    r0                               ; force null return arguments
                     04  03A5          ret                                      ; unwind stack and goto
                         03A6  ;
```

```
                        03A6  ; return control handler
                        03A6  ;
                        03A6  unwind_signal:
                  01    03A6          nop                             ;different than canned_return;
                        03A7  canned_return:
             6D   D5    03A7          tstl    stk_l_cnd_hnd(fp)       ; frame have condition handler?
             2A   13    03A9          beql    100$                    ; if eql then no
                        03AB  ;
                        03AB  ; signal unwind condition by building stach frame
                        03AB  ;
      7E     50   7D    03AB          movq    r0,-(sp)                ; save current return args
      51     5D   D0    03AE          movl    fp,r1                   ; set up call as if called from procedure it
             7E   DC    03B1          movpsl  -(sp)                   ; psl
        D5'AF     9F    03B3          pushab  b^100$                  ; pc
 00000000'8F      DD    03B6          pushl   #ss$_unwind             ; build signal args
             03   DD    03BC          pushl   #3                      ;
                        03BE                                          ; build mechanism args
             7E   7C    03BE          clrq    -(sp)                   ; r0,r1
             7E   D4    03C0          clrl    -(sp)                   ; unwind count
             51   DD    03C2          pushl   r1                      ; target fp
             04   DD    03C4          pushl   #4                      ;
             6E   9F    03C6          pushab  (sp)                    ; set up exception arg list
          18 AE   9F    03C8          pushab  24(sp)                  ;
             50   7C    03CB          clrq    r0                      ; zap input registers
    00 BD    02   FB    03CD          calls   #2,@stk_l_cnd_hnd(fp)   ; call handler
    50    24 AE   7D    03D1          movq    36(sp),r0               ; restore value of return
                  04    03D5  100$:   ret                             ; continue;
                        03D6          .dsabl  lsb
```

```
                                03D6            .sbttl  pli$rvrt_cnd - revert condition handler
                                03D6     ;++
                                03D6     ; pli$rvrt_cnd - revert condition handler
                                03D6     ;
                                03D6     ; functional description:
                                03D6     ;
                                03D6     ; This routine removes a condition handler in the current frame that
                                03D6     ; matchs a specified condition value.
                                03D6     ;
                                03D6     ; inputs:
                                03D6     ;
                                03D6     ;          r0 = condition enable
                                03D6     ;          r1 = fcb address if file related condition or 0 if not file related
                                03D6     ;
                                03D6     ; outputs:
                                03D6     ;
                                03D6     ;          r0,r1 are distroyed.
                                03D6     ;--
                                03D6     pli$rvrt_cnd::
                 7E     52  7D  03D6            movq    r2,-(sp)                     ; save a register
                        6D  D5  03D9            tstl    stk_l_cnd_hnd(fp)            ; is there a condition handler?
                        2E  13  03DB            beql    30$                          ; if eql then no
        52    50    1D  03  EF  03DD            extzv   #3,#29,r0,r2                 ; get condition value less severity
            53    F4    AD  9E  03E2            movab   stk_l_cnd_lst(fp),r3         ; address list of control blocks
                 53     63  D0  03E6     15$:   movl    (r3),r3                      ; get next block
                        20  13  03E9            beql    30$                          ; if eql then done
                   08   A3  D5  03EB            tstl    cnd_l_addr(r3)               ; reverted?
                        F6  13  03EE            beql    15$                          ; if eql then yes
     52     04 A3       1D  03  ED  03F0        cmpzv   #3,#29,cnd_l_enabl(r3),r2; match on this block?
                        EE  12  03F6            bneq    15$                          ; if neq then no
        06 A3     0000'8F   B1  03F8            cmpw    #<<pli$_error>@-16>,cnd_l_enabl+2(r3); pl1 condition?
                        06  12  03FE            bneq    20$                          ; if neq then no
             0C A3      51  D1  0400            cmpl    r1,cnd_l_arg(r3)             ; fcb match?
                        E0  12  0404            bneq    15$                          ; if neq then continue
                   08   A3  D4  0406     20$:   clrl    cnd_l_addr(r3)               ; release from use
                        DB  11  0409            brb     15$                          ;get next block
                 52     8E  7D  040B     30$:   movq    (sp)+,r2                     ; restore
                            05  040E            rsb
```

```
                        040F  ;++
                        040F  ; pli$bound_check - range check error subroutine
                        040F  ;
                        040F  ; functional description:
                        040F  ;
                        040F  ; This routine is entered on a subscript range check. The action is to signal
                        040F  ; the error. If the users continues program execution is resumed.
                        040F  ;
                        040F  ; inputs:
                        040F  ;
                        040F  ;       0(sp) = address of range check code and return address
                        040F  ;
                        040F  ; outputs:
                        040F  ;
                        040F  ;       none
                        040F  ;--
                        040F  pli$bound_check::
              50 8ED0   040F          popl    r0                      ; get return address
      17'AF   00   FB   0412          calls   #0,b^10$                ; create a stack frame
                   04   0416          ret                             ; never used
                 0000   0417  10$:    .word   0                       ;
         51   80   9A   0419          movzbl  (r0)+,r1                ; fetch index in table
      10 AD   50   D0   041C          movl    r0,stk_l_pc(fp)         ; zap return address
    FBDB CF41      DD   0420          pushl   subscript_table[r1]     ;
 00000000'GF   01   FB  0425          calls   #1,g^lib$signal         ;
                   04   042C          ret
```

```
                              042D          .sbttl  pli$resignal - cause resignal of most recent condition
                              042D      ;++
                              042D      ; pli$resignal - resignal most recent condition
                              042D      ;
                              042D      ; functional description:
                              042D      ;
                              042D      ; This routine searches the stack for a condition stack frame and if found
                              042D      ; causes that condition to be resignaled.
                              042D      ;
                              042D      ; inputs:
                              042D      ;
                              042D      ;       none
                              042D      ;
                              042D      ; outputs:
                              042D      ;
                              042D      ;       none
                              042D      ;--
                      001C    042D          .entry  pli$resignal,^m<r2,r3,r4>
             50   5D   D0     042F      3$:     movl    fp,r0                       ; search for a condition
                00B4   30     0432              bsbw    find_cnd_ap                 ; look for a condition stack
                  16   13     0435              beql    10$                         ; if eql then none
          0C AD   50   D1     0437              cmpl    r0,stk_l_fp(fp)             ; next fram is the condition fram?
                  06   13     043B              beql    b^5$                        ; yes, prepare the return address
                              043D      ;
     10 AD    EF AF    9E     043D              movab   b^3$,stk_l_pc(fp)           ; no. pop fram and keep search
                              0442      ;
                       04     0442              ret                                 ; return to pli$resignal
                              0443      ;
     10 A0   4E'AF     9E     0443      5$:     movab   b^30$,stk_l_pc(r0)          ; force return here
     10 AD   4D'AF     9E     0448              movab   b^10$,stk_l_pc(fp)          ; force return to here, instead of
                              044D                                                  ; the caller, so skip the rest of
                              044D                                                  ; this condition handler
                       04     044D      10$:    ret
                              044E      ;
                              044E      ; trapped return
                              044E      ;
          10 AD 8ED0          044E      30$:    popl    stk_l_pc(fp)                ; restore system's stack key
          04 AD 8ED0          0452              popl    stk_l_psl(fp)               ; restore system's psl
                  51   D4     0456              clrl    r1                          ;
 50    00000000'8F     D0     0458              movl    #ss$_resignal,r0            ; resignal condition
                       04     045F              ret                                 ; continue
```

```
                    0460        .sbttl  pli$oncode - get most recent condition name
                    0460 ;++
                    0460 ; pli$oncode - get most recent condition name
                    0460 ;
                    0460 ; functional description:
                    0460 ;
                    0460 ; This routine is a built-in-function that returns a fixed bin(31) value
                    0460 ; which is the name of the most recent condition.  The object is to
                    0460 ; obtain as much information as possible about the condition.
                    0460 ;
                    0460 ; inputs:
                    0460 ;
                    0460 ;       pl1 stack frame.
                    0460 ;
                    0460 ; outputs:
                    0460 ;
                    0460 ;       r0 = condition name value or 0 if no condition is active
                    0460 ;--
               001C 0460        .entry  pli$oncode,^m<r2,r3,r4>
         50  5D  D0 0462        movl    fp,r0                       ; start with this frame
                    0465 ;
                    0465 ; find the argument list of the last signal
                    0465 ;
           0081  30 0465        bsbw    find_cnd_ap                 ; use common routine
             71  13 0468        beql    50$                         ; if eql then none found
                    046A ;
                    046A ; begin search for more information
                    046A ;
      52   04 A1 D0 046A        movl    chf$l_sigarglst(r1),r2      ; address signal arguments
         53   62 D0 046E 10$:   movl    (r2),r3                     ; get count of arguments
      50   04 A2 D0 0471        movl    chf$l_sig_name(r2),r0       ; get actual name
 0000'8F   06 A2 B1 0475        cmpw    chf$l_sig_name+2(r2),#<pli$_error@-16>; pl1 code?
             5E  12 047B        bneq    50$                         ; if neq then return
00000000'8F  50  D1 047D        cmpl    r0,#pli$_endfile            ; endfile condition?
             55  13 0484        beql    50$                         ; yes, return it
00000000'8F  50  D1 0486        cmpl    r0,#pli$_endpage            ; endpage condition?
             4C  13 048D        beql    50$                         ; yes, return it
00000000'8F  50  D1 048F        cmpl    r0,#pli$_finish             ; finish condition?
             43  13 0496        beql    50$                         ; yes, return it
                    0498 ;
                    0498 ; the most recent condition was a pl1 defined condition
                    0498 ;
                    0498 ; PL/I condition of some sort
                    0498 ;
         52  04  C0 0498        addl    #4,r2                       ; address first name
         53  02  C2 049B        subl    #2,r3                       ; remove pc/psl from count
00000000'8F  62  D1 049E 20$:   cmpl    (r2),#pli$_rmsf             ; rmsf condition?
             1E  13 04A5        beql    30$                         ; yes, ignore it
00000000'8F  62  D1 04A7        cmpl    (r2),#pli$_rmsr             ; rmsr condition?
             15  13 04AE        beql    30$                         ; yes, ignore it
00000000'8F  62  D1 04B0        cmpl    (r2),#pli$_ioerror          ; ioerror condition?
             0C  13 04B7        beql    30$                         ; yes, ignore it
00000000'8F  62  D1 04B9        cmpl    (r2),#pli$_filename         ; filename condition?
             03  13 04C0        beql    30$                         ; yes, ignore it
         50  62  D0 04C2        movl    (r2),r0                     ; save most recent condition name
         52  04  C0 04C5 30$:   addl    #4,r2                       ; point past name
         53  02  C2 04C8        subl    #2,r3                       ; adjust count for name and arg count
```

```
          0E   15 04CB             bleq    50$              ; if args runout then done
     51   82   D0 04CD             movl    (r2)+,r1         ; fetch arg count
     53   51   C2 04D0             subl    r1,r3            ; remove this condition name's args
          06   15 04D3             bleq    50$              ; if done then no more
  52    6241   DE 04D5             moval   (r2)[r1],r2      ; look to next
          C3   11 04D9             brb     20$              ;
               04 04DB     50$:    ret                      ; done
```

```
                          04DC          .sbttl  pli$oncndargs - fetch ap of most recent contition
                          04DC      ;++
                          04DC      ; pli$oncndargs - fetch ap of most recent condition
                          04DC      ;
                          04DC      ; functional description:
                          04DC      ;
                          04DC      ; This routine is a built-in-function that returns a pointer which is the
                          04DC      ; ap of the most recent condition.
                          04DC      ; If no condition is active then the pointer returned is null.
                          04DC      ;
                          04DC      ; inputs:
                          04DC      ;
                          04DC      ;       pl1 stack frame
                          04DC      ;
                          04DC      ; outputs:
                          04DC      ;
                          04DC      ;       r0 = return pointer value.
                          04DC      ;--
                    001C  04DC          .entry  pli$oncndarg,^m<r2,r3,r4>
          50    5D    D0  04DE          movl    fp,r0                       ; start with this frame
                06    10  04E1          bsbb    find_cnd_ap
                03    13  04E3          beql    10$                         ; if eql then none found
          50    51    D0  04E5          movl    r1,r0                       ; return value
                      04  04E8      10$:    ret                             ;
                          04E9      ;
                          04E9      ; subroutine to find the ap of the most recent condition
                          04E9      ;
                          04E9      ; r0 = fp to start search so that the search may be continued
                          04E9      ;
                          04E9      find_cnd_ap:
51   FDD4 CF          9E  04E9          movab   w^norm_signal,r1            ; address normal signal
52   FC8E CF          9E  04EE          movab   w^error_signal,r2           ; address error signal
53   FEAF CF          9E  04F3          movab   w^unwind_signal,r3          ;address unwind signal
54   FB9C CF          9E  04F8          movab   w^pli$optmain_hnd,r4        ; address options main handler
  50    0C A0         D0  04FD      10$:    movl    stk_l_fp(r0),r0         ; address next frame
60    10    03        0D  0501          probew  #3,#stk_l_pc,(r0)           ; frame accessible?
                17    13  0505          beql    20$                         ; if eql then end of stack found
   10 A0        51    D1  0507          cmpl    r1,stk_l_pc(r0)             ; this frame return to the pl1 handler?
                14    13  050B          beql    30$                         ; if eql then this is it
   10 A0        52    D1  050D          cmpl    r2,stk_l_pc(r0)             ;this frame called by error signal?
                0E    13  0511          beql    30$                         ;if eql, then this is it
   10 A0        53    D1  0513          cmpl    r3,stk_l_pc(r0)             ;this frame called by goto (unwind)?
                08    13  0517          beql    30$                         ; if eql then this is it
          60    54    D1  0519          cmpl    r4,stk_l_cnd_hnd(r0)        ; this frame's handler the default handler?
                DF    12  051C          bneq    10$                         ; if neq then search again
                50    D4  051E      20$:    clrl    r0                      ; set eql condition codes
                      05  0520          rsb                                 ; return failure
  51    08 A0         D0  0521      30$:    movl    stk_l_ap(r0),r1         ; address ap of the caller's frame
                          0525                                              ; condition codes also set
                      05  0525          rsb                                 ; return
```

```
                                  0526               .sbttl  pli$onfile - onfile bif support
                                  0526       ;++
                                  0526       ; pli$onfile - onfile bif support routine
                                  0526       ;
                                  0526       ; functional description:
                                  0526       ;
                                  0526       ; This routine returns a character varying string which is the file name
                                  0526       ; of the most recent file condition.
                                  0526       ;
                                  0526       ; inputs:
                                  0526       ;
                                  0526       ;       0(ap) = 2;
                                  0526       ;       4(ap) = size of string
                                  0526       ;       8(ap) = address of the string
                                  0526       ;--
                            003C  0526               .entry pli$onfile,^m<r2,r3,r4,r5>
                  50    5D   D0   0528               movl    fp,r0                          ; start with this frame
                                  052B       ;
                                  052B       ; look for the next signal
                                  052B       ;
                        BC   10   052B       10$:    bsbb    find_cnd_ap                    ; locate the current ap
                        42   13   052D               beql    30$                            ; if eql then none
              52     04 A1   D0   052F               movl    chf$l_sigarglst(r1),r2         ; check for pl1 type
         0000'8F     06 A2   B1   0533               cmpw    chf$l_sig_name+2(r2),#<pli$_error@-16>; pl1 type?
                        F0   12   0539               bneq    10$                            ; if neq then no - continue search
00000000'8F    04 A2    1D   03 ED 053B              cmpzv   #3,#29,chf$l_sig_name(r2),#<pli$_key@-3>; file type?
                        E4   1A   0545               bgtru   10$                            ; if no - continue search
              52     08 A1   D0   0547               movl    chf$l_mcharglst(r1),r2         ; address mechanism arguments
              53     0C A2   D0   054B               movl    chf$l_mch_savr0(r2),r3         ; fetch fcb address if any
                        20   13   054F               beql    30$                            ; if eql then no file name
              51     42 A3   9E   0551               movab   fcb_b_ident_nam(r3),r1         ; use identifier name
              50     40 A3   3C   0555               movzwl  fcb_w_ident_len(r3),r0         ;
                        16   13   0559               beql    30$                            ; if eql then no name
              04 AC     50   B1   055B       20$:    cmpw    r0,4(ap)                       ; in range?
                        04   15   055F               bleq    25$                            ; if leq then yes
              50     04 AC   3C   0561               movzwl  4(ap),r0                       ; get return size
              52     08 AC   D0   0565       25$:    movl    8(ap),r2                       ; address target
                 82     50   B0   0569               movw    r0,(r2)+                       ; load size
            62   61     50   28   056C               movc3   r0,(r1),(r2)                   ; return string
                        04   0570                    ret
                     08 BC   B4   0571       30$:    clrw    @8(ap)                         ; return null string
                        04   0574                    ret
```

```
                                  0575          .sbttl  pli$onkey - onkey bif support routine
                                  0575  ;++
                                  0575  ; pli$onkey - onkey bif support routine
                                  0575  ;
                                  0575  ; functional description:
                                  0575  ;
                                  0575  ; This routine returns the key associated with a key condition
                                  0575  ;
                                  0575  ; inputs:
                                  0575  ;
                                  0575  ;       0(ap) = 2;
                                  0575  ;       4(ap) = size of return string
                                  0575  ;       8(ap) = address of the return string
                                  0575  ;
                                  0575  ; --
                            003C  0575          .entry pli$onkey,^m<r2,r3,r4,r5>
                 50    5D    D0   0577          movl    fp,r0                         ; begin search with this frame
                                  057A  ;
                                  057A  ; search for a condition on the stack
                                  057A  ;
                    FF6C     30   057A  10$:    bsbw    find_cnd_ap                   ; find the last condition ap
                       41    13   057D          beql    30$                           ; if eql then no condition found
             52    04  A1    D0   057F          movl    chf$l_sigarglst(r1),r2        ; check for pl1 type
        0000'8F    06  A2    B1   0583          cmpw    chf$l_sig_name+2(r2),#<pli$_error@-16>; pl1 type?
                       EF    12   0589          bneq    10$                           ; if neg then no - continue search
00000000'8F    04 A2   1D    03   ED   058B     cmpzv   #3,#29,chf$l_sig_name(r2),#<pli$_key@-3>; key type?
                       E3    12   0595          bneq    10$                           ; if no - continue search
             50    08  A1    D0   0597          movl    chf$l_mcharglst(r1),r0        ; fetch the secondary arguments
             51    0C  A0    D0   059B          movl    chf$l_mch_savr0(r0),r1        ; get fcb address
                       1F    13   059F          beql    30$                           ; if eql then no key error
             50    10  A0    D0   05A1          movl    chf$l_mch_savr1(r0),r0        ; get string address
                       19    13   05A5          beql    30$                           ; if eql then none
                51     80    3C   05A7          movzwl  (r0)+,r1                      ; get size of string and address it
             04 AC     51    D1   05AA          cmpl    r1,4(ap)                      ; string fit?
                       04    15   05AE          bleq    15$                           ; if leq then yes
             51    04  AC    D0   05B0          movl    4(ap),r1                      ; use passed max size
             52    08  AC    D0   05B4  15$:    movl    8(ap),r2                      ; address target
                82     51    B0   05B8          movw    r1,(r2)+                      ; insert size
          62    60     51    28   05BB          movc3   r1,(r0),(r2)                  ; return string
                       04         05BF          ret
                 08    BC    B4   05C0  30$:    clrw    @8(ap)                        ; return null string
                       04         05C3          ret
```

```
                          05C4         .sbttl  pli$io_error
                          05C4  ;++
                          05C4  ; pli$io_error - signal I/O related error
                          05C4  ;
                          05C4  ; functional description:
                          05C4  ;
                          05C4  ; This routine is used by the runtime I/O routines to signal an error.
                          05C4  ; inputs:
                          05C4  ;        (ap) - 3 or 4
                          05C4  ;        4(ap) - condition value
                          05C4  ;        8(ap) - error code
                          05C4  ;        12(ap) - fcb addr
                          05C4  ;        16(ap) - address of onkey value
                          05C4  ; outputs:
                          05C4  ;        the error message arguments are pushed on the stack and the
                          05C4  ;        condition is signaled.
                          05C4  ;--
                          05C4
                    0FFC  05C4         .entry  pli$io_error,^m<r2,r3,r4,r5,r6,r7,r8,r9,r10,r11>
                 53   D4  05C6         clrl    r3                              ;set no parms
         52   0C AC   D0  05C8         movl    12(ap),r2                       ;fcb passed?
                 03   12  05CC         bneq    5$                              ;if neq, yes
               009D   31  05CE         brw     100$                            ;if eql, no
         0C A2   08   CA  05D1  5$:    bicl    #atr_m_recur,fcb_l_attr(r2) ;clear recursion
00000000'8F   08 AC   D1  05D5         cmpl    8(ap),#pli$_rmsf                ;rms fab error?
                 0A   12  05DD         bneq    10$                             ;in neq, no
            00B2 C2   DD  05DF         pushl   <fcb_b_fab+fab$l_stv>(r2) ;push stv
            00AE C2   DD  05E3         pushl   <fcb_b_fab+fab$l_sts>(r2) ;push sts
                 1C   11  05E7         brb     25$                             ;cont
00000000'8F   08 AC   D1  05E9  10$:   cmpl    8(ap),#pli$_rmsr                ;rms rab error?
                 08   12  05F1         bneq    20$                             ;if neq, no
              6E A2   DD  05F3         pushl   <fcb_b_rab+rab$l_stv>(r2) ;push stv
              6A A2   DD  05F6         pushl   <fcb_b_rab+rab$l_sts>(r2) ;push sts
                 0A   11  05F9         brb     25$                             ;cont
              08 AC   D5  05FB  20$:   tstl    8(ap)                           ;error code specified?
                 08   13  05FE         beql    30$                             ;if eql, no
                 00   DD  0600         pushl   #0                              ;set no fao args
              08 AC   DD  0602         pushl   8(ap)                           ;set error code
              53   02 C0  0605  25$:   addl    #2,r3                           ;update parm count
      3E 0C A2   17   E0  0608  30$:   bbs     #atr_v_string,fcb_l_attr(r2),37$
      54   00F9 C2   9A  060D          movzbl  <fcb_b_nam+nam$b_rsl>(r2),r4 ;get length of rsa
                 07   12  0612         bneq    31$                             ;if neq, cont
      54   0101 C2   9A  0614          movzbl  <fcb_b_nam+nam$b_esl>(r2),r4 ;get length of esa
                 12   13  0619         beql    35$                             ;if eql, cont
            012E C2   9F  061B  31$:   pushab  fcb_b_esa(r2)                   ;set expanded string addr
              7E   54 D0  061F         movl    r4,-(sp)                        ;set len
                 02   DD  0622         pushl   #2                              ;set number of fao args
        00000000'8F   DD  0624         pushl   #pli$_filename                  ;set message
              53   04 C0  062A         addl    #4,r3                           ;update parm count
              42 A2   9F  062D  35$:   pushab  fcb_b_ident_nam(r2)             ;set up file name descr
         7E   40 A2   3C  0630         movzwl  fcb_w_ident_len(r2),-(sp) ;
                 02   DD  0634         pushl   #2                              ;set number of fao args
              53   03 C0  0636         addl    #3,r3                           ;update parm count
00000000'8F   04 AC   D1  0639         cmpl    4(ap),#pli$_error               ;error condition?
                 0C   12  0641         bneq    40$                             ;if neq, no
        00000000'8F   DD  0643         pushl   #pli$_ioerror                   ;set io error to print file name
                 53   D6  0649         incl    r3                              ;update parm count
```

```
             00  DD  064B   987 37$:    pushl   #0                  ;set number of fao args for error
             53  D6  064D   988         incl    r3                  ;update parm count
          04 AC  DD  064F   989 40$:    pushl   4(ap)               ;set pl1 condition
             53  D6  0652   990         incl    r3                  ;update parm count
    50    0C AC  D0  0654   991         movl    12(ap),r0           ;set fcb addr
             51  D4  0658   992         clrl    r1                  ;zap other argument
       04    6C  D1  065A   993         cmpl    (ap),#4             ;onkey passed?
             04  12  065D   994         bneq    50$                 ;if neq, no, cont
    51    10 AC  D0  065F   995         movl    16(ap),r1           ;set onkey address
 00000000'GF 53  FB  0663   996 50$:    calls   r3,g^lib$signal     ;signal the condition
       50    01  D0  066A   997         movl    #1,r0               ;enuf messages
                 04  066D   998         ret                         ;return
             00  DD  066E   999 100$:   pushl   #0                  ;set no fao args
          08 AC  DD  0670  1000         pushl   8(ap)               ;set error code
             00  DD  0673  1001         pushl   #0                  ;set no fao args
          04 AC  DD  0675  1002         pushl   4(ap)               ;set error condition
             50  7C  0678  1003         clrq    r0                  ;set no secondary arguments
 00000000'GF 04  FB  067A  1004         calls   #4,g^lib$signal     ;signal error
       50    01  D0  0681  1005         movl    #1,r0               ;
                 04  0684  1006         ret                         ;
                     0685  1007
```

```
                              0685
                              0685  ;++
                              0685  ; pli$$stream_hnd
                              0685  ; this routine is the condition handler for pli$getfile and pli$putfile.
                              0685  ; it is jsb'd to from an entry mask that lies within the file control
                              0685  ; block. This vectoring is necessary, in order to get the address of the
                              0685  ; file control block.  After calculating the address of the file control
                              0685  ; block, this routine clears the recursion bit in it, so i/o may be performed
                              0685  ; on the file in the users on-unit. Finally, it sets the return address
                              0685  ; for the frame of the get or put to be the address of a return statement
                              0685  ;
                              0685  ; inputs:
                              0685  ;       0(sp) - address of byte following jsb instruction in fcb.
                              0685  ; outputs:
                              0685  ;       none
                              0685  ;--
                              0685
                              0685  pli$$stream_hnd::
       50   00000000'8F   D0  0685          movl    #ss$_resignal,r0                ;assume resignal
 52    6E   000001B6 8F   C3  068C          subl3   #fcb_l_cndaddr+4,(sp),r2        ;get fcb address
      6A 0C A2       03   E5  0694          bbcc    #atr_v_recur,fcb_l_attr(r2),30$ ;clear recursion bit
        6E   01AE C2      9E  0699          movab   fcb_l_condit(r2),(sp)           ;get addr of handler
             52   5D      D0  069E          movl    fp,r2                           ;start with this frame
        52    0C A2       D0  06A1  10$:    movl    stk_l_fp(r2),r2                 ;address next frame
      62   10     03      0D  06A5          probew  #3,#stk_l_pc,(r2)               ;frame accessible?
             58   13          06A9          beql    30$                             ;if eql then end of stack found, return
        62        6E      D1  06AB          cmpl    (sp),stk_l_cnd_hnd(r2)          ;this frame's handler us?
                  F1      12  06AE          bneq    10$                             ;if neq, no, look at next frame
                  62      D4  06B0          clrl    stk_l_cnd_hnd(r2)               ;we aren't needed anymore
        52    04 AC       D0  06B2          movl    chf$l_sigarglst(ap),r2          ;address arg list
             51   62      D0  06B6          movl    chf$l_sig_args(r2),r1           ;if exception handled force
                              06B9                                                  ;continuation at establisher's saved pc
   00000000'8F   04 A2    D1  06B9          cmpl    chf$l_sig_name(r2),#ss$_roprand;check for roprand
                  41      12  06C1          bneq    20$                             ;if neq, resignal
               10 AD      DD  06C3          pushl   stk_l_pc(fp)                    ;save system pc/psl
               04 AD      DD  06C6          pushl   stk_l_psl(fp)                   ;
     10 AD   FC A241      D0  06C9          movl    chf$l_sig_args-4(r2)[r1],stk_l_pc(fp) ;set original pc/psl
      04 AD    6241       D0  06CF          movl    chf$l_sig_args(r2)[r1],stk_l_psl(fp) ;
      00000000'8F         DD  06D4          pushl   #pli$_cnverr                    ;set conversion error subcode
                  7E      D4  06DA          clrl    -(sp)                           ;
      00000000'8F         DD  06DC          pushl   #pli$_error                     ;set error condition code
             50   7C          06E2          clrq    r0                              ;
   00000000'GF    03      FB  06E4          calls   #3,g^lib$signal                 ;and signal pli error
        50     00'8F      9A  06EB          movzbl  #ss$_continue,r0                ;continue after the stream I/O statement
             04 AD      8ED0  06EF          popl    stk_l_psl(fp)                   ;restore system pc/psl
             10 AD      8ED0  06F3          popl    stk_l_pc(fp)                    ;
             51   62      D0  06F7          movl    chf$l_sig_args(r2),r1           ;if exception handled force
                              06FA                                                  ;continuation at establisher's saved pc
 FC A241    00000703'EF   9E  06FA          movab   30$,chf$l_sig_args-4(r2)[r1];(execute a ret upon continuation)
                  04          0703  30$:    ret                                     ;return
                              0704
                              0704  20$:
               10 AD      DD  0704          pushl   stk_l_pc(fp)                    ;save system pc/psl
               04 AD      DD  0707          pushl   stk_l_psl(fp)                   ;
     10 AD   FC A241      D0  070A          movl    chf$l_sig_args-4(r2)[r1],stk_l_pc(fp);set original pc/psl
      04 AD    6241       D0  0710          movl    chf$l_sig_args(r2)[r1],stk_l_psl(fp);
                  53      DD  0715          pushl   r3
```

```
                  51   02    C2  0717                subl    #2,r1
                  53   51    D0  071A                movl    r1,r3
                     6241    DD  071D       25$:     pushl   chf$l_sig_args(r2)[r1]
                    FA 51    F5  0720                sobgtr  r1,25$
   50   04 A2   03   00      EF  0723                extzv   #0,#3,chf$l_sig_name(r2),r0
     6E    03   00   50      F0  0729                insv    r0,#0,#3,(sp)
                       50    7C  072E                clrq    r0
        00000000'GF    53    FB  0730                calls   r3,g^lib$signal
                       53  8ED0  0737                popl    r3
                  04 AD    8ED0  073A                popl    stk_l_psl(fp)
                  10 AD    8ED0  073E                popl    stk_l_pc(fp)
             50   00'8F      9A  0742                movzbl  #ss$_continue,r0
                  51   62    D0  0746                movl    chf$l_sig_args(r2),r1
  FC A241   00000752'EF      9E  0749                movab   40$,chf$l_sig_args-4(r2)[r1]
                             04  0752       40$:     ret
                                 0753                .end
```

PLI
Sym

CBL
CBL
CBL
CBL
CBL
CBL
CBL
CBL
CBL
INI
LIB
PLI
PLI
PLI
PLI
PLI
PLI
PLI
PLI
PLI
PLI
PLI
STK
STK
STK
STK
STK
STK
STK
STK
STK
STS
SYS
SYS

PSE

.
$AB
PLI
_PL

Pha

Ini
Com
Pas
Sym
Pas

```
ANYCOND                 = 00000004               FINISH_SIGNAL_RETURN         000001C0 R      02
ATR_M_RECUR             = 00000008               HARDWARE_TABLE               00000058 R      02
ATR_V_RECUR             = 00000003               INTOVF                     = 00000008
ATR_V_STRING            = 00000017               LIB$SIGNAL                   ********    X   02
CANNED_RETURN             000003A7 R      02     NAM$B_ESL                  = 0000000B
CHF$L_MCHARGLST         = 00000008               NAM$B_RSL                  = 00000003
CHF$L_MCH_FRAME         = 00000004               NORM_EXIT                    0000018D R      02
CHF$L_MCH_SAVR0         = 0000000C               NORM_SIGNAL                  000002C1 R      02
CHF$L_MCH_SAVR1         = 00000010               PLI$$STREAM_HND              00000685 RG     02
CHF$L_SIGARGLST         = 00000004               PLI$$TERM_PROG               ********    X   02
CHF$L_SIG_ARGS          = 00000000               PLI$BOUND_CHECK              0000040F RG     02
CHF$L_SIG_NAME          = 00000004               PLI$CND_HND                  000001C6 RG     02
CND_K_LENGTH              00000014               PLI$DEF_HND                  0000009C RG     02
CND_L_ADDR                00000008               PLI$GOTO                     0000035F RG     02
CND_L_ARG                 0000000C               PLI$IO_ERROR                 000005C4 RG     02
CND_L_ENABL               00000004               PLI$NONLOC_GOTO              0000035F RG     02
CND_L_FLAGS               00000010               PLI$NONLOC_RET               00000330 RG     02
CND_L_LINK                00000000               PLI$ONCNDARG                 000004DC RG     02
DEF_HND_COMMON            0000009E R      02     PLI$ONCODE                   00000460 RG     02
END_OF_PAGE               0000010F R      02     PLI$ONFILE                   00000526 RG     02
ERROR_SIGNAL              00000180 R      02     PLI$ONKEY                    00000575 RG     02
FAB$L_STS               = 00000008               PLI$OPTMAIN_HND              00000098 RG     02
FAB$L_STV               = 0000000C               PLI$OPTMAIN_RET              0000034C RG     02
FATAL_ERROR               00000323 R      02     PLI$PUTFILE_R6               ********    X   02
FATAL_EXCEPTION           00000321 R      02     PLI$PUT_END_R6               ********    X   02
FCB_B_ENVIR               000001C2               PLI$RESIGNAL                 0000042D RG     02
FCB_B_ESA                 0000012E               PLI$RVRT_CND                 000003D6 RG     02
FCB_B_EXTRA               0000003D               PLI$STOP_PROG                ********    X   02
FCB_B_FAB                 000000A6               PLI$_ANYCOND                 ********    X   02
FCB_B_IDENT               00000040               PLI$_CNVERR                  ********    X   02
FCB_B_IDENT_NAM           00000042               PLI$_ENDFILE                 ********    X   02
FCB_B_NAM                 000000F6               PLI$_ENDPAGE                 ********    X   02
FCB_B_NUMKCBS             0000003C               PLI$_ERROR                   ********    X   02
FCB_B_RAB                 00000062               PLI$_FILENAME                ********    X   02
FCB_C_LEN                 000001C2               PLI$_FINISH                  ********    X   02
FCB_C_STRLEN              00000034               PLI$_IOERROR                 ********    X   02
FCB_L_ATTR                0000000C               PLI$_KEY                     ********    X   02
FCB_L_BUF                 00000014               PLI$_RMSF                    ********    X   02
FCB_L_BUF_END             00000018               PLI$_RMSR                    ********    X   02
FCB_L_BUF_PT              0000001C               PLI$_SUBRANGE                ********    X   02
FCB_L_CNDADDR             000001B2               PLI$_SUBRANGE1               ********    X   02
FCB_L_CONDIT              000001AE               PLI$_SUBRANGE2               ********    X   02
FCB_L_DTTR                00000010               PLI$_SUBRANGE3               ********    X   02
FCB_L_ERROR               00000008               PLI$_SUBRANGE4               ********    X   02
FCB_L_KCB                 00000038               PLI$_SUBRANGE5               ********    X   02
FCB_L_NEXT                00000000               PLI$_SUBRANGE6               ********    X   02
FCB_L_PREVIOUS            00000004               PLI$_SUBRANGE7               ********    X   02
FCB_L_PRN                 00000034               PLI$_SUBRANGE8               ********    X   02
FCB_Q_RFA                 00000020               PLI$_SUBSTR2                 ********    X   02
FCB_W_COLUMN              0000002E               PLI$_SUBSTR3                 ********    X   02
FCB_W_IDENT_LEN           00000040               PLI$_ZERODIV                 ********    X   02
FCB_W_LINE                00000030               RAB$L_STS                  = 00000008
FCB_W_LINESIZE            0000002A               RAB$L_STV                  = 0000000C
FCB_W_PAGE                00000032               SEARCH_CND_LIST              000001CB R      02
FCB_W_PAGESIZE            0000002C               SIGNAL_ERROR                 0000012D R      02
FCB_W_REVISION            00000028               SIZ...                     = 00000001
FIND_CND_AP               000004E9 R      02     SS$_ACCVIO                   ********    X   02
```

PLI
VAX

Sym
Pse
Cro
Ass

The
633
The
250
13

Mac

_$2
_$2
TOT

126

The

MAC

```
SS$_CONTINUE                     ********    X    02
SS$_DEBUG                        ********    X    02
SS$_DECOVF                       ********    X    02
SS$_FLTDIV                       ********    X    02
SS$_FLTDIV_F                     ********    X    02
SS$_FLTOVF                       ********    X    02
SS$_FLTOVF_F                     ********    X    02
SS$_FLTUND                       ********    X    02
SS$_FLTUND_F                     ********    X    02
SS$_INTDIV                       ********    X    02
SS$_INTOVF                       ********    X    02
SS$_OPCCUS                       ********    X    02
SS$_OPCDEC                       ********    X    02
SS$_RADRMOD                      ********    X    02
SS$_RESIGNAL                     ********    X    02
SS$_ROPRAND                      ********    X    02
SS$_SUBRNG                       ********    X    02
SS$_UNWIND                       ********    X    02
STK_L_AP                         00000008
STK_L_ARG_LIST                   FFFFFFF8
STK_L_CND_HND                    00000000
STK_L_CND_LST                    FFFFFFF4
STK_L_DISPLAY                    FFFFFFFC
STK_L_FP                         0000000C
STK_L_PC                         00000010
STK_L_PSL                        00000004
STK_L_REGS                       00000014
SUBSCRIPT_TABLE                  00000000 R       02
SYS$EXIT                         ********   GX    02
UNWIND_SIGNAL                    000003A6 R       02
ZERODIV                        = 00000000

                                         +----------------+
                                         ! Psect synopsis !
                                         +----------------+

PSECT name                      Allocation          PSECT No.  Attributes
----------                      ----------          ---------  ----------
.  ABS  .                       00000000  (     0.)  00 (  0.)  NOPIC   USR   CON   ABS   LCL NOSHR NOEXE NORD   NOWRT NOVEC BYTE
$ABS$                           FFFFFFFC  (     0.)  01 (  1.)  NOPIC   USR   CON   ABS   LCL NOSHR   EXE   RD     WRT NOVEC BYTE
_PLI$CODE                       00000753  (  1875.)  02 (  2.)    PIC   USR   CON   REL   LCL   SHR   EXE   RD   NOWRT NOVEC LONG

                                     +------------------------+
                                     ! Performance indicators !
                                     +------------------------+

Phase                     Page faults    CPU Time       Elapsed Time
-----                     -----------    --------       ------------
Initialization                     13    00:00:00.07    00:00:02.15
Command processing                 74    00:00:00.47    00:00:04.70
Pass 1                            240    00:00:09.18    00:00:33.68
Symbol table sort                   0    00:00:01.00    00:00:02.38
Pass 2                            189    00:00:02.77    00:00:09.34
Symbol table output                17    00:00:00.13    00:00:00.31
Psect synopsis output               2    00:00:00.02    00:00:00.09
Cross-reference output              0    00:00:00.00    00:00:00.00
Assembler run totals              535    00:00:13.64    00:00:52.65
```

```
The working set limit was 1350 pages.
53371 bytes (105 pages) of virtual memory were used to buffer the intermediate code.
There were 40 pages of symbol table space allocated to hold 713 non-local and 75 local symbols.
1081 source lines were read in Pass 1, producing 43 object records in Pass 2.
19 pages of virtual memory were used to define 18 macros.

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

Macro library name                       Macros defined
------------------                       --------------
_$255$DUA28:[PLIRTL.OBJ]PLIRTMAC.MLB;1                5
_$255$DUA28:[SYSLIB]STARLET.MLB;2                    10
TOTALS (all libraries)                               15

695 GETS were required to define 15 macros.

There were no errors, warnings or information messages.

MACRO/ENABLE=SUPPRESSION/DISABLE=TRACEBACK/LIS=LIS$:PLICONDIT/OBJ=OBJ$:PLICONDIT MSRC$:PLICONDIT/UPDATE=(ENH$:PLICONDIT)+LIB$:PLIRTM
```

0306 AH-BT13A-SE DIGITAL EQUIPMENT CORPORATION
VAX/VMS V4.0 CONFIDENTIAL AND PROPRIETARY

PLICASERR
LIS

PLICONDIT
LIS

DATDEF
SDL

FILEDEF
IN

PLIRTDEF
MAR

PLICHRSTR
LIS

PLICHAR
LIS

PLICLOSE
LIS

PLIRTL

PLIFILDSP
IN

PLIBYTSIZ
LIS

PLIRTL
MAP

ENVCODES
IN

PLIBIT
LIS

0307 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY

PLIDELETE
LIS

PLICONVRT
LIS

PLIDATA
LIS

PLICONTRL
LIS

PLIDATE
LIS

PLICUTPIC
LIS

PLIENVIR
LIS