```
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
```

```
_$2

Sym
---
PLI
PLI
PLI
PLI
PLI
PLI
PLI
PLI
PLI
PLI
PLI
PLI
PLI
PLI
PLI
PLI
PLI

PLI
PLI
PLI

PLI
PLI
PLI
PLI
PLI
PLI
PLI

PLI
PLI
PLI
PLI

PLI
PLI
PLI
PLI
PLI
PLI
PLI
PLI
PLI
```

FILEID**PLICONDIT

```
PPPPPPPP    LL          IIIIII     CCCCCCCC   000000   NN      NN  DDDDDDDD     IIIIII   TTTTTTTTTT
PPPPPPPP    LL          IIIIII     CCCCCCCC   000000   NN      NN  DDDDDDDD     IIIIII   TTTTTTTTTT
PP      PP  LL            II      CC        00    00  NN      NN  DD      DD     II         TT
PP      PP  LL            II      CC        00    00  NN      NN  DD      DD     II         TT
PP      PP  LL            II      CC        00    00  NNNN    NN  DD      DD     II         TT
PP      PP  LL            II      CC        00    00  NNNN    NN  DD      DD     II         TT
PPPPPPPP    LL            II      CC        00    00  NN  NN  NN  DD      DD     II         TT
PPPPPPPP    LL            II      CC        00    00  NN  NN  NN  DD      DD     II         TT
PP          LL            II      CC        00    00  NN    NNNN  DD      DD     II         TT
PP          LL            II      CC        00    00  NN    NNNN  DD      DD     II         TT
PP          LL            II      CC        00    00  NN      NN  DD      DD     II         TT          ....
PP          LL            II      CC        00    00  NN      NN  DD      DD     II         TT          ....
PP          LLLLLLLLLL  IIIIII     CCCCCCCC   000000   NN      NN  DDDDDDDD   IIIIII       TT          ....
PP          LLLLLLLLLL  IIIIII     CCCCCCCC   000000   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

```
0000     1          .title  pli$condit -  condition handler routines
0000     2          .ident  /1-008/                                         ; Edit DSB1008
0000     3                                                                  ; Edit DSB1007
0000     4                                                                  ; Edit CGN1006
0000     5                                                                  ; Edit CGN1005
0000     6                                                                  ; Edit CGN1004
0000     7                                                                  ; Edit CGN1003
0000     8                                                                  ; Edit WHM1002
0000     9 ;
0000    10 ;
0000    11 ;*****************************************************************************
0000    12 ;*                                                                           *
0000    13 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                 *
0000    14 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000    15 ;*  ALL RIGHTS RESERVED.                                                     *
0000    16 ;*                                                                           *
0000    17 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    18 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    19 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    20 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    21 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    22 ;*  TRANSFERRED.                                                             *
0000    23 ;*                                                                           *
0000    24 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    25 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    26 ;*  CORPORATION.                                                             *
0000    27 ;*                                                                           *
0000    28 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    29 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    30 ;*                                                                           *
0000    31 ;*                                                                           *
0000    32 ;*****************************************************************************
0000    33 ;
0000    34 ;++
0000    35 ; facility:
0000    36 ;
0000    37 ;       VAX/VMS PL1 runtime library.
0000    38 ;
0000    39 ; abstract:
0000    40 ;
0000    41 ;       This module contains the pl1 runtime routines for condition
0000    42 ;       handling and non-local goto processing.
0000    43 ;
0000    44 ; author: r. heinen 18-jan-1979
0000    45 ;
0000    46 ; Modifications:
0000    47 ;
0000    48 ;
0000    49 ;       1-002   Bill Matthews   29-September-1982
0000    50 ;
0000    51 ;               Invoke macros $defdat and rtshare instead of $defopr and share.
0000    52 ;
0000    53 ;       1-003   Chip Nylander   25-January-1983
0000    54 ;
0000    55 ;               Fix the ONCODE() bif.  It was blown to bits by a previous
0000    56 ;               edit, and did not return the correct results for most cases.
0000    57 ;               The new, and correct, behavior is:
```

```
0000     58 ;
0000     59 ;               1) For PLI$_ENDFILE, PLI$_ENDPAGE, PLI$_FINISH, and all
0000     60 ;                  non-PLI conditions, return the primary condition code.
0000     61 ;                  Note that FIXEDOVERFLOW, OVERFLOW, UNDERFLOW, and
0000     62 ;                  ZERODIVIDE are not signalled as PL/I conditions.
0000     63 ;
0000     64 ;               2) for all other conditions (currently ERROR, KEY,
0000     65 ;                  UNDEFINEDFILE, and VAXCONDITION) search the signal
0000     66 ;                  argument list and return the last (most specific)
0000     67 ;                  condition, but ignore all instances of PLI$_FILENAME,
0000     68 ;                  PLI$_RMSF, PLI$_RMSR, and PLI$_IOERROR because they
0000     69 ;                  are "uninteresting".
0000     70 ;
0000     71 ;       1-004   Chip Nylander   07-February-1983
0000     72 ;
0000     73 ;               Make floating faults visible to ON units.
0000     74 ;
0000     75 ;       1-005   Chip Nylander   22-February-1983
0000     76 ;
0000     77 ;               Prevent spurious error messages when a VAX condition is
0000     78 ;               signalled and not handled:
0000     79 ;               when calling LIB$SIGNAL from the OPTIONS(MAIN) condition
0000     80 ;               to make PLI$_ERROR the primary error, don't pass the saved
0000     81 ;               psl/pc to LIB$SIGNAL for those system errors that don't
0000     82 ;               want it.
0000     83 ;
0000     84 ;       1-006   Chip Nylander   17-March-1983
0000     85 ;
0000     86 ;               Make SIGNAL_ERROR a local symbol rather than a global one.
0000     87 ;
0000     88 ;       1-007   Dave Blickstein 14-February-1984
0000     89 ;
0000     90 ;               Fixup norm_exit in PL$DEF_HND to resignal to the system
0000     91 ;               handler when PLIG-ERROR is received with I severity.
0000     92 ;               It was signalling FINISH before passing the error on.
0000     93 ;
0000     94 ;       1-008   Dave Blickstein 23-May-1984
0000     95 ;
0000     96 ;               Changed PLI$OPTMAIN_HND to resignal SS$_DEBUG.  This was
0000     97 ;               done so that PL/I programs can be ^Y-ed and examined with
0000     98 ;               the debugger with the DEBUG command.  Previously, SS$_DEBUG
0000     99 ;               would be resignalled like any other error and a traceback
0000    100 ;               would occur followed by an image exit.
0000    101 ;--
0000    102
0000    103 ;
0000    104 ; external definitions
0000    105 ;
0000    106
0000    107         $chfdef                         ; define condition handler offsets
0000    108         $defcnd                         ; define condition handler block
0000    109         $deffcb                         ; define runtime file control block
0000    110         $defstk                         ; define runtime stack
0000    111         $fabdef                         ; define fab offsets
0000    112         $rabdef                         ; define rab offsets
0000    113         $namdef                         ; define nam offsets
0000    114         $sfdef                          ; define stack offset
```

```
                                0000   115
                                0000   116 ;
                                0000   117 ; local definitions
                                0000   118 ;
                     00000000   0000   119         zerodiv = 0
                     00000004   0000   120         anycond = 4
                     00000008   0000   121         intovf  = 8
                                0000   122
                                0000   123
                                0000   124         rtshare
                                0000   125 ;
                                0000   126 ; table of subscript range message numbers
                                0000   127 ;
                                0000   128 subscript_table:
                     00000000'  0000   129         .long   pli$_subrange
                     00000000'  0004   130         .long   pli$_subrange1
                     00000000'  0008   131         .long   pli$_subrange2
                     00000000'  000C   132         .long   pli$_subrange3
                     00000000'  0010   133         .long   pli$_subrange4
                     00000000'  0014   134         .long   pli$_subrange5
                     00000000'  0018   135         .long   pli$_subrange6
                     00000000'  001C   136         .long   pli$_subrange7
                     00000000'  0020   137         .long   pli$_subrange8
00000000 00000000 00000000 00000000     0024   138         .long   0,0,0,0,0,0,0,0,0,0,0,0
00000000 00000000 00000000 00000000     0034
         00000000 00000000 00000000     0044
                     00000000'  0050   139         .long   pli$_substr2
                     00000000'  0054   140         .long   pli$_substr3
                                0058   141
                                0058   142 ;
                                0058   143 ; table of hardware-raised errors that need the PSL and PC returned to DCL
                                0058   144 ; for FAO formatting.
                                0058   145 ;
                                0058   146 hardware_table:
                     00000000'  0058   147         .long   ss$_intovf
                     00000000'  005C   148         .long   ss$_intdiv
                     00000000'  0060   149         .long   ss$_fltovf
                     00000000'  0064   150         .long   ss$_fltovf_f
                     00000000'  0068   151         .long   ss$_fltund
                     00000000'  006C   152         .long   ss$_fltund_f
                     00000000'  0070   153         .long   ss$_fltdiv
                     00000000'  0074   154         .long   ss$_fltdiv_f
                     00000000'  0078   155         .long   ss$_decovf
                     00000000'  007C   156         .long   ss$_subrng
                     00000000'  0080   157         .long   ss$_accvio
                     00000000'  0084   158         .long   ss$_roprand
                     00000000'  0088   159         .long   ss$_radrmod
                     00000000'  008C   160         .long   ss$_opcdec
                     00000000'  0090   161         .long   ss$_opccus
                     00000000   0094   162         .long   0                       ; must be last
                                0098   163
```

```
                     0098   165          .sbttl  pli$def_hnd - pl1 default condition handler
                     0098   166 ;++
                     0098   167 ; pli$def_hnd - pl1 default condition handler
                     0098   168 ;
                     0098   169 ; functional description:
                     0098   170 ;
                     0098   171 ; This routine is the condition handler for all pl1 main programs.
                     0098   172 ; The action is to search the normal on unit list in the same way
                     0098   173 ; as the standard handler. If a condition handler was found then the
                     0098   174 ; condition is continued. If no condition handler was found then error is signaled.
                     0098   175 ;
                     0098   176 ; inputs:
                     0098   177 ;
                     0098   178 ;       condition argument list
                     0098   179 ;
                     0098   180 ; outputs:
                     0098   181 ;
                     0098   182 ;       none, but the program may be aborted.
                     0098   183 ; Two entry points are provided so that the remainder of the runtime
                     0098   184 ; can distinguish the highest level options main handler
                     0098   185 ;--
           0FFC      0098   186          .entry  pli$optmain_hnd,^m<r2,r3,r4,r5,r5,r6,r7,r8,r9,r10,r11>
        02   11      009A   187          brb     def_hnd_common          ; use common code for handler
           0FFC      009C   188          .entry  pli$def_hnd,^m<r2,r3,r4,r5,r6,r7,r8,r9,r10,r11>;
                     009E   189 def_hnd_common:
                     009E   190 ;
                     009E   191 ; search this frames condition list for an on unit
                     009E   192 ;
              012A   30      009E   193          bsbw    search_cnd_list         ; look for a handler in this frame
        52    04 AC  D0      00A1   194          movl    chf$l_sigarglst(ap),r2  ; address the signal arg list
                     00A5   195 ;
                     00A5   196 ; if an on unit was found then continue
                     00A5   197 ;
  00000000'8F    50  D1      00A5   198          cmpl    r0,#ss$_continue        ; handler found?
                 01  12      00AC   199          bneq    10$                     ; if neq then perform default action
                     04      00AE   200          ret                             ; continue after user handler
                     00AF   201
                     00AF   202 ;
                     00AF   203 ; perform default action based on the condition value
                     00AF   204 ;
                     00AF   205 10$:
    0000'8F    06 A2  B1      00AF   206          cmpw    chf$l_sig_name+2(r2),#<pli$_errora-16>; pl1 defined?
                 1F  12      00B5   207          bneq    20$                     ; signal pl1 error condition
50  04 A2    0A   03  EF      00B7   208          extzv   #3,#10,chf$l_sig_name(r2),r0; get message number
      0000'8F    50  B1      00BD   209          cmpw    r0,#<pli$_finish&^xfff>a-3; finish?
                 3F  13      00C2   210          beql    23$                     ; if eql then continue
                     00C4   211          case    r0,<-
                     00C4   212                  signal_error,-          ; never signaled
                     00C4   213                  end_of_page,-           ; end page
                     00C4   214                  signal_error,-          ; end file
                     00C4   215                  signal_error,-          ; unknown file
                     00C4   216                  signal_error,-          ; key
                     00C4   217                  norm_exit-              ; error signaled, so exit
                     00C4   218                  >
                 57  11      00D4   219          brb     signal_error            ; all others cause error signal
                     00D6   220 ;
                     00D6   221 ; non pl1 condition
```

```
                                00D6   222 ;
          00000000'8F    04 A2   D1   00D6   223 20$:    cmpl    chf$l_sig_name(r2),#ss$_fltund; floating underflow trap?
                            29   13   00DE   224        beql    25$                           ; if eql then ignore
          00000000'8F    04 A2   D1   00E0   225        cmpl    chf$l_sig_name(r2),#ss$_fltund_f; floating underflow fault?
                            1F   13   00E8   226        beql    25$                           ; if eql then ignore
          00000000'8F    04 A2   D1   00EA   227        cmpl    chf$l_sig_name(r2),#ss$_debug   ; if activate debugger
                            03   12   00F2   228        bneq    22$                           ;
                          00C9   31   00F4   229        brw     finish_signal_return          ; then resignal
 00000000'8F    04 A2    1D     03   ED   00F7   230 22$:    cmpzv   #3,#29,chf$l_sig_name(r2),#<ss$_unwind@-3>; unwind?
                            2A   12   0101   231        bneq    signal_error                  ; all others get default handling
                  00000000'EF   16   0103   232 23$:    jsb     pli$$term_prog                ; shut off program exit control
                50    0000'8F   3C   0109   233 25$:    movzwl  #ss$_continue,r0              ; continue
                                04   010E   234        ret
                                     010F   235 ;
                                     010F   236 ; end of page
                                     010F   237 ;
                                     010F   238 end_of_page:
                   52    08 AC    D0   010F   239        movl    chf$l_mcharglst(ap),r2  ;get mech args
                   5C    0C A2    D0   0113   240        movl    chf$l_mch_savr0(r2),ap  ;set fcb addr
                            50    D4   0117   241        clrl    r0                      ;set no line
                      51    01    D0   0119   242        movl    #1,r1                   ;set to put page
                      52    00    7D   011C   243        movq    #0,r2                   ;set no skip, no options
                            54    D4   011F   244        clrl    r4                      ;set no format
                  00000000'GF    16   0121   245        jsb     g^pli$putfile_r6        ;do put page
                  00000000'GF    17   0127   246        jmp     g^pli$put_end_r6        ;finish it off and return
                                     012D   247 ;
                                     012D   248 ; signal error condition
                                     012D   249 ;
                                     012D   250 signal_error:
                      10 AD    DD   012D   251        pushl   stk_l_pc(fp)            ;save system pc/psl
                      04 AD    DD   0130   252        pushl   stk_l_psl(fp)           ;
                   50    62    D0   0133   253        movl    chf$l_sig_args(r2),r0   ;get size of arg list
          10 AD    FC A240    D0   0136   254        movl    chf$l_sig_args-4(r2)[r0],stk_l_pc(fp) ;set original pc/psl
             04 AD    6240    D0   013C   255        movl    chf$l_sig_args(r2)[r0],stk_l_psl(fp) ;
             53    FF13 CF    DE   0141   256        moval   hardware_table,r3       ; set to top of hardware error table
                            63   D5   0146   257 10$:    tstl    (r3)                    ; any more?
                            0C   13   0148   258        beql    30$                     ; no, so throw away pc/psl
                   83    04 A2   D1   014A   259        cmpl    chf$l_sig_name(r2),(r3)+; need pc/psl?
                            F6   12   014E   260        bneq    10$                     ; if not, loop back to check again
                53    50    02   C1   0150   261        addl3   #2,r0,r3                ;get number of args for signal
                            06   11   0154   262        brb     50$                     ; rejoin common
                      53    50   D0   0156   263 30$:    movl    r0,r3                   ;copy number of args for signal
                      50    02   C2   0159   264        subl    #2,r0                   ;set number of args to copy less pc/psl
                         6240   DD   015C   265 50$:    pushl   chf$l_sig_args(r2)[r0]  ;copy args to stack
                      FA 50     F5   015F   266        sobgtr  r0,50$                  ;
                            00   DD   0162   267        pushl   #0                      ;set no args for error
                   00000000'8F   DD   0164   268        pushl   #pli$_error             ;set error
 50    04 A2    03    00   EF   016A   269        extzv   #0,#3,chf$l_sig_name(r2),r0; get severity of previous condition
    6E    03    00    50   F0   0170   270        insv    r0,#0,#3,(sp)           ; insert as severity of this signal
                            6D   D4   0175   271        clrl    (fp)                    ;disappear our condition handler
                            50   7C   0177   272        clrq    r0                      ;set no secondary arguments
          00000000'GF    53   FB   0179   273        calls   r3,g^lib$signal         ;signal error
                                     0180   274 error_signal:                       ;if we come back, the signal was handled
                      04 AD 8ED0   0180   275        popl    stk_l_psl(fp)           ;restore system pc/psl
                      10 AD 8ED0   0184   276        popl    stk_l_pc(fp)            ;
                   50    00'8F    9A   0188   277        movzbl  #ss$_continue,r0        ;set to continue
                                04   018C   278        ret                             ;return
```

```
                                018D   279 ;
                                018D   280 ; end the program
                                018D   281 ;
                                018D   282 norm_exit:
    04   04 A2   03   00   ED  018D   283         cmpzv   #0,#3,chf$l_sig_name(r2),#4; fatal severity?
                         2B   19  0193   284         blss    finish_signal_return    ; if leq then no
                00000000'GF   16  0195   285         jsb     g^pli$$term_prog        ; shut off exit handler finish condition
                        10 AD  DD  019B   286         pushl   stk_l_pc(fp)            ;save system pc/psl
                        04 AD  DD  019E   287         pushl   stk_l_psl(fp)           ;
                  10 AD   B8'AF  9E  01A1   288         movab   b^10$,stk_l_pc(fp)      ;set unrecognized pc/psl
                        04 AD  DC  01A6   289         movpsl  stk_l_psl(fp)           ;
                00000000'8F    DD  01A9   290         pushl   #pli$_finish            ; signal fi.  h condition
                         6D    D4  01AF   291         clrl    (fp)                    ; remove trouble handler
            000 0000'GF    01   FB  01B1   292         calls   #1,g^lib$signal
                        04 AD 8ED0  01B8   293 10$:     popl    stk_l_psl(fp)           ;restore system pc/psl
                        10 AD 8ED0  01BC   294         popl    stk_l_pc(fp)            ;
                                01C0   295 ;
                                01C0   296 ; print messages via system or next handler
                                01C0   297 ;
                                01C0   298 finish_signal_return:
          50    0000'8F    3C  01C0   299         movzwl  #ss$_resignal,r0        ; get the messages
                           04  01C5   300         ret                             ; just return
```

```
                        01C6   302          .sbttl  pli$cnd_hnd - condition handler for pl1
                        01C6   303 ;++
                        01C6   304 ; pli$cnd_hnd - condition handler for pl1
                        01C6   305 ;
                        01C6   306 ; functional description:
                        01C6   307 ;
                        01C6   308 ; This routine handles conditions for pl1 programs by searching the
                        01C6   309 ; current condition handler list for a control block which specifies
                        01C6   310 ; the same enable. The first control block with a matching enable is
                        01C6   311 ; used to call the associated condition handler routine.
                        01C6   312 ; If no handler is found then the condition is resignaled.
                        01C6   313 ;
                        01C6   314 ; inputs:
                        01C6   315 ;
                        01C6   316 ;       exception stack frame
                        01C6   317 ;       The signaller's r1 is meaningfull on known pl1 conditions.
                        01C6   318 ;       If the condition relates to a file then that r1 is the fcb address
                        01C6   319 ;       of the file.
                        01C6   320 ;
                        01C6   321 ; outputs:
                        01C6   322 ;
                        01C6   323 ;       none
                        01C6   324 ;
                        01C6   325 ; If an error occurs while handling the exception, a fatal exit is taken.
                        01C6   326 ; A signal will not work because control will return here.
                        01C6   327 ;
                        01C6   328 ; The call tc the condition handler is done with the argument list of the condition.
                        01C6   329 ;
                        01C6   330 ;--
                        01C6   331          .enabl  lsb
                001C    01C6   332          .entry  pli$cnd_hnd,^m<r2,r3,r4>
             01   10    01C8   333          bsbb    search_cnd_list         ; search the condition list
                  04    01CA   334          ret                             ; done
```

```
                                01CB    336          .sbttl search_cnd_list
                                01CB    337 ;
                                01CB    338 ; subroutine to search the establisher's condition handler list
                                01CB    339 ;
                                01CB    340 search_cnd_list:
                6D   0321'CF    9E      01CB    341          movab   w^fatal_exception,(fp)  ; set up fatal exception vector
                       7E       7C      01D0    342          clrq    -(sp)                   ; zero stack locals
                       7E       D4      01D2    343          clrl    -(sp)                   ;
                  51    04 AC    D0      01D4    344          movl    chf$l_sigarglst(ap),r1  ; get the signal value
                  52    04 A1    D0      01D8    345          movl    chf$l_sig_name(r1),r2   ;
                     54    61    D0      01DC    346          movl    chf$l_sig_args(r1),r4   ; get argument numbers
                54    FC A144    DE      01DF    347          moval   chf$l_sig_args-4(r1)[r4],r4; address the pc
                  53    08 AC    D0      01E4    348          movl    chf$l_mcharglst(ap),r3  ; address arg list with fp of establisher
                  50    0C A3    D0      01E8    349          movl    chf$l_mch_savr0(r3),r0  ;get fcb addr if its there
                  51    04 A3    D0      01EC    350 1$:      movl    chf$l_mch_frame(r3),r1  ;
                                01F0    351 ;
                                01F0    352 ; search condition list in that frame
                                01F0    353 ;
                  53    F4 A1    9E      01F0    354          movab   stk_l_cnd_lst(r1),r3    ; address condition handler control list
                     53    63    D0      01F4    355 5$:      movl    (r3),r3                 ; get next block
                           03    12      01F7    356          bneq    10$                     ; if eql then done
                              00DB 31      01F9    357          brw     40$                     ; resignal condition
                           08 A3    D5      01FC    358 10$:     tstl    cnd_l_addr(r3)          ; reverted condition handler?
                              F3    13      01FF    359          beql    5$                      ; if eql then yes
                                0201    360 ;
                                0201    361 ; look for fixed overflow handler
                                0201    362 ;
00000001'8F     04 A3    1D    03      ED      0201    363          cmpzv   #3,#29,cnd_l_enabl(r3),#<pli$_zerodiv+8>@-3; PLI fixed overflow?
                           09    12      020B    364          bneq    15$                     ; if neq then no
                              08 AE    D5      020D    365          tstl    intovf(sp)              ; handler found already?
                           04    12      0210    366          bneq    15$                     ; if neq then yes
                        08 AE    53      D0      0212    367          movl    r3,intovf(sp)           ; save for default case
                                0216    368 ;
                                0216    369 ; look for an anycondition handler
                                0216    370 ;
00000000'8F     04 A3    1D    03      ED      0216    371 15$:     cmpzv   #3,#29,cnd_l_enabl(r3),#<pli$_anycond>@-3; PLI any condition?
                           0B    12      0220    372          bneq    20$                     ; if neq then no
                              04 AE    D5      0222    373          tstl    anycond(sp)             ; already found in list?
                           06    12      0225    374          bneq    20$                     ; if neq then yes
                        04 AE    53      D0      0227    375          movl    r3,anycond(sp)          ; save for default condition
                           13    11      022B    376          brb     25$                     ; continue
                                022D    377 ;
                                022D    378 ; look for zero divide handler
                                022D    379 ;
00000000'8F     04 A3    1D    03      ED      022D    380 20$:     cmpzv   #3,#29,cnd_l_enabl(r3),#<pli$_zerodiv>@-3; PLI zero divide
                           07    12      0237    381          bneq    25$                     ; if neq then no
                           6E    D5      0239    382          tstl    zerodiv(sp)             ; this condition enabled already?
                           03    12      023B    383          bneq    25$                     ; if neq then yes
                        6E    53      D0      023D    384          movl    r3,zerodiv(sp)          ; save special dual value case
                                0240    385 ;
                                0240    386 ; look at secondary argument if PLI condition
                                0240    387 ;
   00000000'8F    52    10    10      ED      0240    388 25$:     cmpzv   #16,#16,r2,#<<pli$_error>@-16>; pl1 defined condition signaled?
                           42    13      0249    389          beql    29$                     ; if eql then yes
                     52    04 A3    D1      024B    390          cmpl    cnd_l_enabl(r3),r2      ; enabled?
                           5B    13      024F    391          beql    30$                     ; if eql then signal condition
         00000000'8F    04 A3    D1      0251    392          cmpl    cnd_l_enabl(r3),#ss$_fltovf; enabled for overflow?
```

```
                    09   12  0259   393          bneq    26$                             ; no - check next fault
          00000000'8F    52  D1  025B   394          cmpl    r2,#ss$_fltovf_f                ; yes - is this an overflow fault?
                    48   13  0262   395          beql    30$                             ; if yes signal condition
       00000000'8F  04 A3  D1  0264   396 26$:     cmpl    cnd_l_enabl(r3),#ss$_fltund; enabled for underflow?
                    09   12  026C   397          bneq    27$                             ; no - check next fault
          00000000'8F    52  D1  026E   398          cmpl    r2,#ss$_fltund_f                ; yes - is this an underflow fault?
                    35   13  0275   399          beql    30$                             ; if yes signal condition
       00000000'8F  04 A3  D1  0277   400 27$:     cmpl    cnd_l_enabl(r3),#ss$_fltdiv; enabled for divide?
                    09   12  027F   401          bneq    28$                             ; no - go find next handler
          00000000'8F    52  D1  0281   402          cmpl    r2,#ss$_fltdiv_f                ; yes - is this a divide fault?
                    22   13  0288   403          beql    30$                             ; if yes signal condition
                FF67    31  028A   404 28$:     brw     5$                              ; look for next handler
          7E    1D    03  EF  028D   405 29$:     extzv   #3,#29,r2,-(sp)                 ; get internal bits
       8E  04 A3  1D    03  ED  0292   406          cmpzv   #3,#29,cnd_l_enabl(r3),(sp)+; match in pl1 bits?
                    F0   12  0298   407          bneq    28$                             ; if neq then no
00000000'8F  04 A3  1D    03  ED  029A   408          cmpzv   #3,#29,cnd_l_enabl(r3),#<pli$_keya-3>; file type?
                    06   1A  02A4   409          bgtru   30$                             ; if gtru then no additional arg
             50  0C A3  D1  02A6   410          cmpl    cnd_l_arg(r3),r0                ; same secondary argument?
                    DE   12  02AA   411          bneq    28$                             ; if neq then no
                             02AC   412 ;
                             02AC   413 ; call with r1 as establisher's frame pointer
                             02AC   414 ;
                    6D   D4  02AC   415 30$:     clrl    (fp)                            ; disable fatal exception vector
                 04 AD   DD  02AE   416          pushl   stk_l_psl(fp)                   ; save system's psl
                 10 AD   DD  02B1   417          pushl   stk_l_pc(fp)                    ; save system's stack key
              10 AD   64   D0  02B4   418          movl    (r4),stk_l_pc(fp)               ; set pc/psw to original
           04 AD   04   A4  B0  02B8   419          movw    4(r4),stk_l_psl(fp)             ; change psw only
              08 B3   6C   FA  02BD   420          callg   (ap),@cnd_l_addr(r3)            ; call handler with our arg list
                             02C1   421 norm_signal:                                    ; condition handler stack key
           6D   0321'CF   9E  02C1   422          movab   w^fatal_exception,(fp)          ; setup a condition handler
                 10 AD 8ED0  02C6   423          popl    stk_l_pc(fp)                    ; restore system's stack key
                 04 AD 8ED0  02CA   424          popl    stk_l_psl(fp)                   ; restore system's psl
        50   0000'8F   3C  02CE   425 35$:     movzwl  #ss$_continue,r0                ; set up to continue
             5E   0C   C0  02D3   426          addl    #12,sp                          ; purge stack
                    05   02D6   427          rsb                                     ; and do so
                             02D7   428 ;
                             02D7   429 ; no condition handler found - check defaults
                             02D7   430 ;
             53   6E   D0  02D7   431 40$:     movl    zerodiv(sp),r3                  ; get zero divide handler
                    1B   13  02DA   432          beql    45$                             ; if eql then none
00000000'8F         52   D1  02DC   433          cmpl    r2,#ss$_fltdiv                  ; floating divide trap?
                    39   13  02E3   434          beql    60$                             ; if yes then merge
00000000'8F         52   D1  02E5   435          cmpl    r2,#ss$_fltdiv_f                ; floating divide fault?
                    30   13  02EC   436          beql    60$                             ; if yes then merge
00000000'8F         52   D1  02EE   437          cmpl    r2,#ss$_intdiv                  ; integer divide?
                    27   13  02F5   438          beql    60$                             ; if eql then merge
          53   08 AE   D0  02F7   439 45$:     movl    intovf(sp),r3                   ; integer overflow?
                    12   13  02FB   440          beql    50$                             ; if eql then no handler found
00000000'8F         52   D1  02FD   441          cmpl    r2,#ss$_decovf                  ; decimal overflow?
                    18   13  0304   442          beql    60$                             ; if eql then handle condition
00000000'8F         52   D1  0306   443          cmpl    r2,#ss$_intovf                  ; integer overflow?
                    0F   13  030D   444          beql    60$                             ; if eql then handle
          53   04 AE   D0  030F   445 50$:     movl    anycond(sp),r3                  ; any condition handler?
                    09   12  0313   446          bneq    60$                             ; if neq then handle
        50   0000'8F   3C  0315   447 55$:     movzwl  #ss$_resignal,r0                ; set up to continue signal
             5E   0C   C0  031A   448          addl    #12,sp                          ; purge stack
                    05   031D   449          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   461          .sbttl  pli$nonloc_ret - non-local return processing
                     0330   462 ;++
                     0330   463 ; pli$nonloc_ret - non-local return
                     0330   464 ;
                     0330   465 ; functional description:
                     0330   466 ;
                     0330   467 ; This routine is entered via a jmp from a pl1 program when a non-local
                     0330   468 ; return occurs. The action is to unwind the call stack.
                     0330   469 ; The unwind is done in a manner similar to that of SYS$UNWIND.
                     0330   470 ;
                     0330   471 ; At each stack frame that is passed, the SS$_UNWIND condition is signaled.
                     0330   472 ;
                     0330   473 ; No provision is made for protecting r0/r1 from restoration by a call mask.
                     0330   474 ; PL/I does not save r0/r1
                     0330   475 ;
                     0330   476 ; inputs:
                     0330   477 ;
                     0330   478 ;       r0/r1 return value if any
                     0330   479 ;       r2 = number of frames to skip
                     0330   480 ;
                     0330   481 ; outputs:
                     0330   482 ;
                     0330   483 ;       none
                     0330   484 ;--
                     0330   485 pli$nonloc_ret::                        ; not a call entry
 7E   50   7D        0330   486          movq    r0,-(sp)                ; save return value and regs
 50   5D   D0        0333   487          movl    fp,r0                   ; copy current fp
                     0336   488 ;
                     0336   489 ; check for proper frame
                     0336   490 ;
10 A0   A7'AF   9E   0336   491 10$:     movab   b^canned_return,stk_l_pc(r0); force return in this module
   50   0C A0   D0   033B   492          movl    stk_l_fp(r0),r0         ; link to next frame on stack
 60   10   03   0D   033F   493          probew  #3,#stk_l_pc,(r0)       ; frame accessible?
           DE   13   0343   494          beql    fatal_error             ; if eql then insuff frames
        EE 52   F5   0345   495          sobgtr  r2,10$                  ; continue until done
                     0348   496
                     0348   497 ;
                     0348   498 ; proper frame found
                     0348   499 ;
 50   8E   7D        0348   500          movq    (sp)+,r0                ; restore return value
           04        034B   501          ret                             ; unwind stack and goto
```

```
                        034C   503 ;++
                        034C   504 ; pli$optmain_ret - return in options main block
                        034C   505 ;
                        034C   506 ; functional description:
                        034C   507 ;
                        034C   508 ; This routine performs a return inside an options main block
                        034C   509 ; The action is to return unless this is the outermost options main block.
                        034C   510 ; If it the outermost block then a STOP is done.
                        034C   511 ;
                        034C   512 ; inputs:
                        034C   513 ;
                        034C   514 ;       none
                        034C   515 ;
                        034C   516 ; outputs:
                        034C   517 ;
                        034C   518 ;       none - r0,r1 are preserved for function returns
                        034C   519 ;--
                        034C   520 pli$optmain_ret::
                        034C   521 ;
                        034C   522 ; if this frame is the options main frame then STOP otherwise just return
                        034C   523 ;
                FD48 CF 9F 034C   524         pushab  w^pli$optmain_hnd       ; address options main handler
                6D   8E D1 0350   525         cmpl    (sp)+,stk_l_cnd_hnd(fp) ; this frame the outermost?
                     09 12 0353   526         bneq    5$                      ; if neq then return
                        0355   527 ;
                        0355   528 ; cause a stop to occur
                        0355   529 ;
                     50 DD 0355   530         pushl   r0                      ; set return value for exit code
      00000000'GF    01 FB 0357   531         calls   #1,g^pli$stop_prog      ; execute STOP
                        04 035E   532 5$:     ret                             ; continue
```

```
                        035F    534          .sbttl  pli$nonloc_goto - non-local goto processing
                        035F    535 ;++
                        035F    536 ; pli$nonloc_goto - non-local goto
                        035F    537 ;
                        035F    538 ; functional description:
                        035F    539 ;
                        035F    540 ; This routine is entered via a jmp from a pl1 program when a non-local
                        035F    541 ; goto occurs. The action is to unwind the call stack.
                        035F    542 ; The unwind is done in a manner similar to that of SYS$UNWIND.
                        035F    543 ;
                        035F    544 ; At each stack frame that is passed, the SS$_UNWIND condition is signaled.
                        035F    545 ;
                        035F    546 ; inputs:
                        035F    547 ;
                        035F    548 ;       r0 = new pc
                        035F    549 ;       r1 = fp of target frame
                        035F    550 ;
                        035F    551 ; outputs:
                        035F    552 ;
                        035F    553 ;       none
                        035F    554 ;--
                        035F    555 pli$nonloc_goto::
                        035F    556 pli$goto::                                  ; not a call entry
                        035F    557          .enabl  lsb
5D   51   D1            035F    558          cmpl    r1,fp                      ; is this the frame?
     02   12            0362    559          bneq    10$                        ; if neq then no
     60   17            0364    560          jmp     (r0)                       ; goto new pc
                        0366    561 ;
                        0366    562 ; this goto is outside this stack frame
                        0366    563 ;
     50   DD            0366    564 10$:     pushl   r0                         ; save new pc
  50   5D   D0          0368    565          movl    fp,r0                      ; copy current fp
                        036B    566 ;
                        036B    567 ; check for proper frame
                        036B    568 ;
0C A0   51   D1         036B    569 20$:     cmpl    r1,stk_l_fp(r0)            ; is the next frame the last?
        2E   13         036F    570          beql    40$                        ; if eql then yes
7E  FF4C CF   9E        0371    571          movab   w^norm_signal,-(sp)        ;address normal signal ret addr
  8E   10 A0   D1       0376    572          cmpl    stk_l_pc(r0),(sp)+         ;does this frame point there?
           1?   13      037A    573          beql    25$                        ;if eql, yes, special case
7E  FE00 CF   9E        037C    574          movab   w^error_signal,-(sp)       ;address error signal ret addr
  8E   10 AC   D1       0381    575          cmpl    stk_l_pc(r0),(sp)+         ;does this frame point there?
           07   13      0385    576          beql    25$                        ;if eql, yes, special case
10 A0   A7'AF   9E      0387    577          movab   b^canned_return,stk_l_pc(r0); force return in this module
           05   11      038C    578          brb     27$                        ;cont
10 A0   A6'AF   9E      038E    579 25$:     movab   b^unwind_signal,stk_l_pc(r0); force return for signal frame
  50   CC A0   D0       0393    580 27$:     movl    stk_l_fp(r0),r0            ; link to next frame on stack
60   10   03   0D       0397    581          probew  #3,#stk_l_pc,(r0)          ; frame accessible?
           86   13      039B    582          beql    fatal_error                ; if eql then insuff frames
           CC   11      039D    583          brb     20$                        ; continue search
                        039F    584 ;
                        039F    585 ; proper frame found
                        039F    586 ;
     10 A0 8ED0         039F    587 40$:     popl    stk_l_pc(r0)               ; force new pc on return
        50   7C         03A3    588          clrq    r0                         ; force null return arguments
             04         03A5    589          ret                                ; unwind stack and goto
                        03A6    590 ;
```

```
                     03A6   591 ; return control handler
                     03A6   592 ;
                     03A6   593 unwind_signal:
                  01 03A6   594         nop                             ;different than canned_return;
                     03A7   595 canned_return:
               6D    D5 03A7   596         tstl    stk_l_cnd_hnd(fp)       ; frame have condition handler?
               2A    13 03A9   597         beql    100$                    ; if eql then no
                     03AB   598 ;
                     03AB   599 ; signal unwind condition by building stach frame
                     03AB   600 ;
            7E    50 7D 03AB   601         movq    r0,-(sp)                ; save current return args
            51    5D D0 03AE   602         movl    fp,r1                   ; set up call as if called from procedure it
               7E DC 03B1   603         movpsl  -(sp)                   ; psl
         D5    AF 9F 03B3   604         pushab  b^100$                  ; pc
00000000'      8F DD 03B6   605         pushl   #ss$_unwind             ; build signal args
               03 DD 03BC   606         pushl   #3                      ;
                     03BE   607                                         ; build mechanism args
               7E 7C 03BE   608         clrq    -(sp)                   ; r0,r1
               7E D4 03C0   609         clrl    -(sp)                   ; unwind count
               51 DD 03C2   610         pushl   r1                      ; target fp
               04 DD 03C4   611         pushl   #4                      ;
               6E 9F 03C6   612         pushab  (sp)                    ; set up exception arg list
            18 AE 9F 03C8   613         pushab  24(sp)                  ;
               50 7C 03CB   614         clrq    r0                      ; zap input registers
      00 BD    02 FB 03CD   615         calls   #2,@stk_l_cnd_hnd(fp)   ; call handler
      50    24 AE 7D 03D1   616         movq    36(sp),r0               ; restore value of return
                  04 03D5   617 100$:   ret                             ; continue;
                     03D6   618         .dsabl  lsb
```

```
                        03D6   620          .sbttl  pli$rvrt_cnd - revert condition handler
                        03D6   621 ;++
                        03D6   622 ; pli$rvrt_cnd - revert condition handler
                        03D6   623 ;
                        03D6   624 ; functional description:
                        03D6   625 ;
                        03D6   626 ; This routine removes a condition handler in the current frame that
                        03D6   627 ; matchs a specified condition value.
                        03D6   628 ;
                        03D6   629 ; inputs:
                        03D6   630 ;
                        03D6   631 ;       r0 = condition enable
                        03D6   632 ;       r1 = fcb address if file related condition or 0 if not file related
                        03D6   633 ;
                        03D6   634 ; outputs:
                        03D6   635 ;
                        03D6   636 ;       r0,r1 are distroyed.
                        03D6   637 ;--
                        03D6   638 pli$rvrt_cnd::
                7E  52  7D  03D6   639          movq    r2,-(sp)                ; save a register
                    6D  D5  03D9   640          tstl    stk_l_cnd_hnd(fp)       ; is there a condition handler?
                    2E  13  03DB   641          beql    30$                     ; if eql then no
        52  50  1D      03  EF  03DD   642          extzv   #3,#29,r0,r2            ; get condition value less severity
              53  F4  AD  9E  03E2   643          movab   stk_l_cnd_lst(fp),r3    ; address list of control blocks
                53    63  D0  03E6   644 15$:     movl    (r3),r3                 ; get next block
                    20  13  03E9   645          beql    30$                     ; if eql then done
                  08  A3  D5  03EB   646          tstl    cnd_l_addr(r3)          ; reverted?
                    F6  13  03EE   647          beql    15$                     ; if eql then yes
    52  04  A3      1D  03  ED  03F0   648          cmpzv   #3,#29,cnd_l_enabl(r3),r2; match on this block?
                    EE  12  03F6   649          bneq    15$                     ; if neq then no
        06  A3  0000'8F  B1  03F8   650          cmpw    #<<pli$_error>a-16>,cnd_l_enabl+2(r3); pl1 condition?
                    06  12  03FE   651          bneq    20$                     ; if neq then no
              0C  A3    51  D1  0400   652          cmpl    r1,cnd_l_arg(r3)        ; fcb match?
                    E0  12  0404   653          bneq    15$                     ; if neq then continue
                  08  A3  D4  0406   654 20$:     clrl    cnd_l_addr(r3)          ; release from use
                    DB  11  0409   655          brb     15$                     ;get next block
            52      8E  7D  040B   656 30$:     movq    (sp)+,r2                ; restore
                        05  040E   657          rsb
```

```
                     040F    659 ;++
                     040F    660 ; pli$bound_check - range check error subroutine
                     040F    661 ;
                     040F    662 ; functional description:
                     040F    663 ;
                     040F    664 ; The routine is entered on a subscript range check. The action is to signal
                     040F    665 ; the error. If the users continues program execution is resumed.
                     040F    666 ;
                     040F    667 ; inputs:
                     040F    668 ;
                     040F    669 ;       0(sp) = address of range check code and return address
                     040F    670 ;
                     040F    671 ; outputs:
                     040F    672 ;
                     040F    673 ;       none
                     040F    674 ;--
                     040F    675 pli$bound_check::
              50 8ED0  040F    676         popl    r0                      ; get return address
    17'AF     00   FB  0412    677         calls   #0,b^10$                ; create a stack frame
                   04  0416    678         ret                             ; never used
                 0000  0417    679 10$:    .word   0                       ;
       51     80   9A  0419    680         movzbl  (r0)+,r1                ; fetch index in table
    10 AD     50   D0  041C    681         movl    r0,stk_l_pc(fp)         ; zap return address
      FBDB CF41   DD  0420    682         pushl   subscript_table[r1]     ;
00000000'GF     01   FB  0425    683         calls   #1,g^lib$signal         ;
                   04  042C    684         ret
```

```
                                042D   686          .sbttl  pli$resignal - cause resignal of most recent condition
                                042D   687 ;++
                                042D   688 ; pli$resignal - resignal most rece ι condition
                                042D   689 ;
                                042D   690 ; functional description:
                                042D   691 ;
                                042D   692 ; This routine searches the stack for a condition stack frame and if found
                                042D   693 ; causes that condition to be resignaled.
                                042D   694 ;
                                042D   695 ; inputs:
                                042D   696 ;
                                042D   697 ;       none
                                042D   698 ;
                                042D   699 ; outputs:
                                042D   700 ;
                                042D   701 ;       none
                                042D   702 ;--
                         001C   042D   703          .entry  pli$resignal,^m<r2,r3,r4>
                    50    5D    D0   042F   704 3$:       movl    fp,r0                   ; search for a condition
                        00B4    30   0432   705           bsbw    find_cnd_ap             ; look for a condition stack
                          16    13   0435   706           beql    10$                     ; if eql then none
                    0C AD 50    D1   0437   707           cmpl    r0,stk_l_fp(fp)         ; next fram is the condition fram?
                          06    13   043B   708           beql    b^5$                    ; yes, prepare the return address
                                043D   709 ;
                10 AD   EF AF   9E   043D   710           movab   b^3$,stk_l_pc(fp)       ; no. pop fram and keep search
                                0442   711 ;
                                04   0442   712           ret                             ; return to pli$resignal
                                0443   713 ;
                10 A0   4E'AF   9E   0443   714 5$:       movab   b^30$,stk_l_pc(r0)      ; force return here
                10 AD   4D'AF   9E   0448   715           movab   b^10$,stk_l_pc(fp)      ; force return to here, instead of
                                044D   716                                               ; the caller, so skip the rest of
                                044D   717                                               ; this condition handler
                                04   044D   718 10$:       ret
                                044E   719 ;
                                044E   720 ; trapped return
                                044E   721 ;
                      10 AD 8ED0   044E   722 30$:       popl    stk_l_pc(fp)            ; restore system's stack key
                      04 AD 8ED0   0452   723           popl    stk_l_psl(fp)           ; restore system's psl
                          51    D4   0456   724           clrl    r1                      ;
            50  00000000'8F     D0   0458   725           movl    #ss$_resignal,r0        ; resignal condition
                                04   045F   726           ret                             ; continue
```

```
                                0460   728          .sbttl  pli$oncode - get most recent condition name
                                0460   729 ;++
                                0460   730 ; pli$oncode - get most recent condition name
                                0460   731 ;
                                0460   732 ; functional description:
                                0460   733 ;
                                0460   734 ; This routine is a built-in-function that returns a fixed bin(31) value
                                0460   735 ; which is the name of the most recent condition.  The object is to
                                0460   736 ; obtain as much information as possible about the condition.
                                0460   737 ;
                                0460   738 ; inputs:
                                0460   739 ;
                                0460   740 ;       pl1 stack frame.
                                0460   741 ;
                                0460   742 ; outputs:
                                0460   743 ;
                                0460   744 ;       r0 = condition name value or 0 if no condition is active
                                0460   745 ;--
                        001C    0460   746          .entry  pli$oncode,^m<r2,r3,r4>
                    50  5D      0462   747          movl    fp,r0                   ; start with this frame
                                0465   748 ;
                                0465   749 ; find the argument list of the last signal
                                0465   750 ;
                          0081  30     0465   751          bsbw    find_cnd_ap             ; use common routine
                            71  13     0468   752          beql    50$                     ; if eql then none found
                                046A   753 ;
                                046A   754 ; begin search for more information
                                046A   755 ;
                        52  04 A1      046A   756          movl    chf$l_sigarglst(r1),r2  ; address signal arguments
                            53  62     046E   757 10$:     movl    (r2),r3                 ; get count of arguments
                        50  04 A2      0471   758          movl    chf$l_sig_name(r2),r0   ; get actual name
                  0000'8F   06 A2      0475   759          cmpw    chf$l_sig_name+2(r2),#<pli$_errora-16>; pl1 code?
                            5E  12     047B   760          bneq    50$                     ; if neq then return
                00000000'8F 50  D1     047D   761          cmpl    r0,#pli$_endfile        ; endfile condition?
                            55  13     0484   762          beql    50$                     ; yes, return it
                00000000'8F 50  D1     0486   763          cmpl    r0,#pli$_endpage        ; endpage condition?
                            4C  13     048D   764          beql    50$                     ; yes, return it
                00000000'8F 50  D1     048F   765          cmpl    r0,#pli$_finish         ; finish condition?
                            43  13     0496   766          beql    50$                     ; yes, return it
                                0498   767 ;
                                0498   768 ; the most recent condition was a pl1 defined condition
                                0498   769 ;
                                0498   770 ; PL/I condition of some sort
                                0498   771 ;
                        52  04  C0     0498   772          addl    #4,r2                   ; address first name
                        53  02  C2     049B   773          subl    #2,r3                   ; remove pc/psl from count
                00000000'8F 62  D1     049E   774 20$:     cmpl    (r2),#pli$_rmsf         ; rmsf condition?
                            1E  13     04A5   775          beql    30$                     ; yes, ignore it
                00000000'8F 62  D1     04A7   776          cmpl    (r2) #pli$_rmsr         ; rmsr condition?
                            15  13     04AE   777          beql    30$                     ; yes, ignore it
                00000000'8F 62  D1     04B0   778          cmpl          pli$_ioerror      ; ioerror condition?
                            0C  13     04B7   779          beql                            ; yes, ignore it
                00000000'8F 62  D1     04B9   780          cmpl        ,#pli$_filename     ; filename condition?
                            03  13     04C0   781          beql    30$                     ; yes, ignore it
                        50  62  D0     04C2   782          movl    (r2),r0                 ; save most recent condition name
                        52  04  C0     04C5   783 30$:     addl    #4,r2                   ; point past name
                        53  02  C2     04C8   784          subl    #2,r3                   ; adjust count for name and arg count
```

```
                0E   15  04CB   785          bleq    50$                     ; if args runout then done
          51    82   D0  04CD   786          movl    (r2)+,r1                ; fetch arg count
          53    51   C2  04D0   787          subl    r1,r3                   ; remove this condition name's args
                06   15  04D3   788          bleq    50$                     ; if done then no more
          52  6241   DE  04D5   789          moval   (r2)[r1],r2             ; look to next
                C3   11  04D9   790          brb     20$                     ;
                     04  04DB   791 50$:     ret                             ; done
```

```
                        04DC   793          .sbttl  pli$oncndargs - fetch ap of most recent contition
                        04DC   794 ;++
                        04DC   795 ; pli$oncndargs - fetch ap of most recent condition
                        04DC   796 ;
                        04DC   797 ; functional description:
                        04DC   798 ;
                        04DC   799 ; This routine is a built-in-function that returns a pointer which is the
                        04DC   800 ; ap of the most recent condition.
                        04DC   801 ; If no condition is active then the pointer returned is null.
                        04DC   802 ;
                        04DC   803 ; inputs:
                        04DC   804 ;
                        04DC   805 ;       pl1 stack frame
                        04DC   806 ;
                        04DC   807 ; outputs:
                        04DC   808 ;
                        04DC   809 ;       r0 = return pointer value.
                        04DC   810 ;--
                   001C 04DC   811          .entry  pli$oncndarg,^m<r2,r3,r4>
          50    5D    D0 04DE   812          movl    fp,r0                   ; start with this frame
                06    10 04E1   813          bsbb    find_cnd_ap
                03    13 04E3   814          beql    10$                     ; if eql then none found
          50    51    D0 04E5   815          movl    r1,r0                   ; return value
                04       04E8   816 10$:     ret                             ;
                         04E9   817 ;
                         04E9   818 ; subroutine to find the ap of the most recent condition
                         04E9   819 ;
                         04E9   820 ; r0 = fp to start search so that the search may be continued
                         04E9   821 ;
                         04E9   822 find_cnd_ap:
     51    FDD4    CF    9E 04E9   823          movab   w^norm_signal,r1        ; address normal signal
     52    FC8E    CF    9E 04EE   824          movab   w^error_signal,r2       ; address error signal
     53    FEAF    CF    9E 04F3   825          movab   w^unwind_signal,r3      ;address unwind signal
     54    FB9C    CF    9E 04F8   826          movab   w^pli$optmain_hnd,r4    ; address options main handler
       50      0C    A0    D0 04FD   827 10$:     movl    stk_l_fp(r0),r0         ; address next frame
     60      10          03    CD 0501   828          probew  #3,#stk_l_pc,(r0)       ; frame accessible?
                         17    13 0505   829          beql    20$             ; if eql then end of stack found
       10    A0          51    D1 0507   830          cmpl    r1,stk_l_pc(r0)         ; this frame return to the pl1 handler?
                         14    13 050B   831          beql    30$             ; if eql then this is it
       10    A0          52    D1 050D   832          cmpl    r2,stk_l_pc(r0)         ;this frame called by error signal?
                         0E    13 0511   833          beql    30$             ;if eql, then this is it
       10    A0          53    D1 0513   834          cmpl    r3,stk_l_pc(r0)         ;this frame called by goto (unwind)?
                         08    13 0517   835          beql    30$             ; if eql then this is it
                60       54    D1 0519   836          cmpl    r4,stk_l_cnd_hnd(r0)    ; this frame's handler the default handler?
                         DF    12 051C   837          bneq    10$             ; if neq then search again
                         50    D4 051E   838 20$:     clrl    r0              ; set eql condition codes
                         05    0520   839          rsb                             ; return failure
       51      08    A0    D0 0521   840 30$:     movl    stk_l_ap(r0),r1         ; address ap of the caller's frame
                               0525   841                                         ; condition codes also set
                         05    0525   842          rsb                             ; return
```

```
                                0526   844          .sbttl  pli$onfile - onfile bif support
                                0526   845 ;++
                                0526   846 ; pli$onfile - onfile bif support routine
                                0526   847 ;
                                0526   848 ; functional description:
                                0526   849 ;
                                0526   850 ; This routine returns a character varying string which is the file name
                                0526   851 ; of the most recent file condition.
                                0526   852 ;
                                0526   853 ; inputs:
                                0526   854 ;
                                0526   855 ;       0(ap) = 2;
                                0526   856 ;       4(ap) = size of string
                                0526   857 ;       8(ap) = address of the string
                                0526   858 ;--
                        003C    0526   859          .entry pli$onfile,^m<r2,r3,r4,r5>
                   50   5D     D0   0528   860          movl    fp,r0                   ; start with this frame
                                052B   861 ;
                                052B   862 ; look for the next signal
                                052B   863 ;
                        BC      10   052B   864 10$:    bsbb    find_cnd_ap             ; locate the current ap
                        42      13   052D   865          beql    30$                     ; if eql then none
                52      04 A1   D0   052F   866          movl    chf$l_sigarglst(r1),r2  ; check for pl1 type
           0000'8F      06 A2   B1   0533   867          cmpw    chf$l_sig_name+2(r2),#<pli$_error@-16>; pl1 type?
                        F0      12   0539   868          bneq    10$                     ; if neq then no - continue search
 00000000'8F    04 A2   1D      03   053B   869          cmpzv   #3,#29,chf$l_sig_name(r2),#<pli$_key@-3>; file type?
                        E4      1A   0545   870          bgtru   10$                     ; if no - continue search
                52      08 A1   D0   0547   871          movl    chf$l_mcharglst(r1),r2  ; address mechanism arguments
                53      0C A2   D0   054B   872          movl    chf$l_mch_savr0(r2),r3  ; fetch fcb address if any
                        20      13   054F   873          beql    30$                     ; if eql then no file name
                51      42 A3   9E   0551   874          movab   fcb_b_ident_nam(r3),r1  ; use identifier name
                50      40 A3   3C   0555   875          movzwl  fcb_w_ident_len(r3),r0  ;
                        16      13   0559   876          beql    30$                     ; if eql then no name
             04 AC      50      B1   055B   877 20$:     cmpw    r0,4(ap)                ; in range?
                        04      15   055F   878          bleq    25$                     ; if leq then yes
                50      04 AC   3C   0561   879          movzwl  4(ap),r0                ; get return size
                52      08 AC   D0   0565   880 25$:     movl    8(ap),r2                ; address target
                   82           50   0569   881          movw    r0,(r2)+                ; load size
             62    61           28   056C   882          movc3   r0,(r1),(r2)            ; return string
                                04   0570   883          ret
                   08 BC        B4   0571   884 30$:     clrw    @8(ap)                  ; return null string
                                04   0574   885          ret
```

```
                                0575   887          .sbttl  pli$onkey - onkey bif support routine
                                0575   888 ;++
                                0575   889 ; pli$onkey - onkey bif support routine
                                0575   890 ;
                                0575   891 ; functional description:
                                0575   892 ;
                                0575   893 ; This routine returns the key associated with a key condition
                                0575   894 ;
                                0575   895 ; inputs:
                                0575   896 ;
                                0575   897 ;       0(ap) = 2;
                                0575   898 ;       4(ap) = size of return string
                                0575   899 ;       8(ap) = address of the return string
                                0575   900 ;
                                0575   901 ; --
                         003C   0575   902          .entry pli$onkey,^m<r2,r3,r4,r5>
                   50    5D    D0   0577   903          movl    fp,r0                   ; begin search with this frame
                                057A   904 ;
                                057A   905 ; search for a condition on the stack
                                057A   906 ;
                      FF6C      30   057A   907 10$:     bsbw    find_cnd_ap             ; find the last condition ap
                         41     13   057D   908          beql    30$                     ; if eql then no condition found
                52    04 A1    D0   057F   909          movl    chf$l_sigarglst(r1),r2  ; check for pl1 type
          0000'8F     06 A2    B1   0583   910          cmpw    chf$l_sig_name+2(r2),#<pli$_error@-16>; pl1 type?
                         EF     12   0589   911          bneq    10$                     ; if neg then no - continue search
00000000'8F    04 A2    1D    03    ED   058B   912          cmpzv   #3,#29,chf$l_sig_name(r2),#<pli$_key@-3>; key type?
                         E3     12   0595   913          bneq    10$                     ; if no - continue search
             50    08 A1    D0   0597   914          movl    chf$l_mcharglst(r1),r0  ; fetch the secondary arguments
             51    0C A0    D0   059B   915          movl    chf$l_mch_savr0(r0),r1  ; get fcb address
                   1F     13   059F   916          beql    30$                     ; if eql then no key error
             50    10 A0    D0   05A1   917          movl    chf$l_mch_savr1(r0),r0  ; get string address
                   19     13   05A5   918          beql    30$                     ; if eql then none
                51 80     3C   05A7   919          movzwl  (r0)+,r1                ; get size of string and address it
          04 AC    51     D1   05AA   920          cmpl    r1,4(ap)                ; string fit?
                   04     15   05AE   921          bleq    15$                     ; if leq then yes
             51    04 AC    D0   05B0   922          movl    4(ap),r1                ; use passed max size
             52    08 AC    D0   05B4   923 15$:     movl    8(ap),r2                ; address target
                82    51     B0   05B8   924          movw    r1,(r2)+                ; insert size
          62    60    51     28   05BB   925          movc3   r1,(r0),(r2)            ; return string
                         04     05BF   926          ret
                   08 BC    B4   05C0   927 30$:     clrw    @8(ap)                  ; return null string
                         04     05C3   928          ret
```

```
                                05C4   930          .sbttl  pli$io_error
                                05C4   931 ;++
                                05C4   932 ; pli$io_error - signal I/O related error
                                05C4   933 ;
                                05C4   934 ; functional description:
                                05C4   935 ;
                                05C4   936 ; This routine is used by the runtime I/O routines to signal an error.
                                05C4   937 ; inputs:
                                05C4   938 ;       (ap) - 3 or 4
                                05C4   939 ;       4(ap) - condition value
                                05C4   940 ;       8(ap) - error code
                                05C4   941 ;       12(ap) - fcb addr
                                05C4   942 ;       16(ap) - address of onkey value
                                05C4   943 ; outputs:
                                05C4   944 ;       the error message arguments are pushed on the stack and the
                                05C4   945 ;       condition is signaled.
                                05C4   946 ;--
                                05C4   947
                           0FFC 05C4   948          .entry  pli$io_error,^m<r2,r3,r4,r5,r6,r7,r8,r9,r10,r11>
                         53   D4 05C6   949          clrl    r3                      ;set no parms
                      52 0C AC   D0 05C8   950          movl    12(ap),r2               ;fcb passed?
                         03      12 05CC   951          bneq    5$                      ;if neq, yes
                       009D      31 05CE   952          brw     100$                    ;if eql, no
                   0C A2         CA 05D1   953 5$:      bicl    #atr_m_recur,fcb_l_attr(r2) ;clear recursion
               00000000'8F 08 AC   D1 05D5   954          cmpl    8(ap),#pli$_rmsf        ;rms fab error?
                         0A      12 05DD   955          bneq    10$                     ;in neq, no
                      00B2 C2    DD 05DF   956          pushl   <fcb_b_fab+fab$l_stv>(r2) ;push stv
                      00AE C2    DD 05E3   957          pushl   <fcb_b_fab+fab$l_sts>(r2) ;push sts
                         0C      11 05E7   958          brb     25$                     ;cont
               00000000'8F 08 AC   D1 05E9   959 10$:     cmpl    8(ap),#pli$_rmsr        ;rms rab error?
                         08      12 05F1   960          bneq    20$                     ;if neq, no
                      6E A2      DD 05F3   961          pushl   <fcb_b_rab+rab$l_stv>(r2) ;push stv
                      6A A2      DD 05F6   962          pushl   <fcb_b_rab+rab$l_sts>(r2) ;push sts
                         0A      11 05F9   963          brb     25$                     ;cont
                      08 AC      D5 05FB   964 20$:     tstl    8(ap)                   ;error code specified?
                         08      13 05FE   965          beql    30$                     ;if eql, no
                         00      DD 0600   966          pushl   #0                      ;set no fao args
                      08 AC      DD 0602   967          pushl   8(ap)                   ;set error code
                   53    02      C0 0605   968 25$:     addl    #2,r3                   ;update parm count
                3E 0C A2   17      E0 0608   969 30$:     bbs     #atr_v_string,fcb_l_attr(r2),37$
                 54 00F9 C2    9A 060D   970          movzbl  <fcb_b_nam+nam$b_rsl>(r2),r4 ;get length of rsa
                         07      12 0612   971          bneq    31$                     ;if neq, cont
                 54 0101 C2    9A 0614   972          movzbl  <fcb_b_nam+nam$b_esl>(r2),r4 ;get length of esa
                         12      13 0619   973          beql    35$                     ;if eql, cont
                      012E C2    9F 061F   974 31$:     pushab  fcb_b_esa(r2)           ;set expanded string addr
                      7E   54    D0 0621   975          movl    r4,-(sp)                ;set len
                         02      DD 0622   976          pushl   #2                      ;set number of fao args
                   00000000'8F    DD 0624   977          pushl   #pli$_filename          ;set message
                   53    04      C0 062A   978          addl    #4,r3                   ;update parm count
                     42 A2      9F 062D   979 35$:     pushab  fcb_b_ident_nam(r2)     ;set up file name descr
                   7E    40 A2   3C 0630   980          movzwl  fcb_w_ident_len(r2),-(sp) ;
                         02      DD 0634   981          pushl   #2                      ;set number of fao args
                   53    03      C0 0636   982          addl    #3,r3                   ;update parm count
               00000000'8F 04 AC   D1 0639   983          cmpl    4(ap),#pli$_error       ;error condition?
                         0C      12 0641   984          bneq    40$                     ;if neq, no
                   00000000'8F    DD 0643   985          pushl   #pli$_ioerror           ;set io error to print file name
                         53      D6 0649   986          incl    r3                      ;update parm count
```

```
                                                B 1
                     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         mov'    #1,r0                   ;
                     04    0684   1006         ret                             ;
                           0685   1007
```

```
                                0685  1009
                                0685  1010 ;++
                                0685  1011 ; pli$$stream_hnd
                                0685  1012 ; this routine is the condition handler for pli$getfile and pli$putfile.
                                0685  1013 ; it is jsb'd to from an entry mask that lies within the file control
                                0685  1014 ; block. This vectoring is necessary, in order to get the address of the
                                0685  1015 ; file control block.  After calculating the address of the file control
                                0685  1016 ; block, this routine clears the recursion bit in it, so i/o may be performed
                                0685  1017 ; on the file in the users on-unit. Finally, it sets the return address
                                0685  1018 ; for the frame of the get or put to be the address of a return statement
                                0685  1019 ;
                                0685  1020 ; inputs:
                                0685  1021 ;       0(sp) - address of byte following jsb instruction in fcb.
                                0685  1022 ; outputs:
                                0685  1023 ;       none
                                0685  1024 ;--
                                0685  1025
                                0685  1026 pli$$stream_hnd::
     50   00000000'8F    D0    0685  1027          movl    #ss$_resignal,r0        ;assume resignal
52   6E    000001B6 8F    C3    068C  1028          subl3   #fcb_l_cndaddr+4,(sp),r2 ;get fcb address
        6A 0C A2    03    E5    0694  1029          bbcc    #atr_v_recur,fcb_l_attr(r2),30$ ;clear recursion bit
         6E   01AE C2    9E    0699  1030          movab   fcb_l_condit(r2),(sp)   ;get addr of handler
              52   5D    D0    069E  1031          movl    fp,r2                   ;start with this frame
           52   0C A2    D0    06A1  1032 10$:     movl    stk_l_fp(r2),r2         ;address next frame
         62   10   03    0D    06A5  1033          probew  #3,#stk_l_pc,(r2)       ;frame accessible?
                   58    13    06A9  1034          beql    30$                     ;if eql then end of stack found, return
              62   6E    D1    06AB  1035          cmpl    (sp),stk_l_cnd_hnd(r2)  ;this frame's handler us?
                   F1    12    06AE  1036          bneq    10$                     ;if neq, no, look at next frame
                   62    D4    06B0  1037          clrl    stk_l_cnd_hnd(r2)       ;we aren't needed anymore
           52   04 AC    D0    06B2  1038          movl    chf$l_sigarglst(ap),r2  ;address arg list
              51   62    D0    06B6  1039          movl    chf$l_sig_args(r2),r1   ;if exception handled force
                         06B9  1040                          ;continuation at establisher's saved pc
00000000'8F   04 A2    D1    06B9  1041          cmpl    chf$l_sig_name(r2),#ss$_roprand;check for roprand
                   41    12    06C1  1042          bneq    20$                     ;if neq, resignal
                10 AD    DD    06C3  1043          pushl   stk_l_pc(fp)            ;save system pc/psl
                04 AD    DD    06C6  1044          pushl   stk_l_psl(fp)           ;
     10 AD   FC A241    D0    06C9  1045          movl    chf$l_sig_args-4(r2)[r1],stk_l_pc(fp) ;set original pc/psl
        04 AD   6241    D0    06CF  1046          movl    chf$l_sig_args(r2)[r1],stk_l_psl(fp) ;
         00000000'8F    DD    06D4  1047          pushl   #pli$_cnverr            ;set conversion error subcode
                   7E    D4    06DA  1048          clrl    -(sp)                   ;
         00000000'8F    DD    06DC  1049          pushl   #pli$_error             ;set error condition code
                   50    7C    06E2  1050          clrq    r0                      ;
    00000000'GF    03    FB    06E4  1051          calls   #3,g^lib$signal         ;and signal pli error
         50   00'8F    9A    06EB  1052          movzbl  #ss$_continue,r0        ;continue after the stream I/O statement
                04 AD 8ED0    06EF  1053          popl    stk_l_psl(fp)           ;restore system pc/psl
                10 AD 8ED0    06F3  1054          popl    stk_l_pc(fp)            ;
              51   62    D0    06F7  1055          movl    chf$l_sig_args(r2),r1   ;if exception handled force
                         06FA  1056                          ;continuation at establisher's saved pc
FC A241   00000703'EF    9E    06FA  1057          movab   30$,chf$l_sig_args-4(r2)[r1];(execute a ret upon continuation)
                   04    0703  1058 30$:     ret                             ;return
                         0704  1059
                         0704  1060 20$:
                10 AD    DD    0704  1061          pushl   stk_l_pc(fp)            ;save system pc/psl
                04 AD    DD    0707  1062          pushl   stk_l_psl(fp)           ;
     10 AD   FC A241    D0    070A  1063          movl    chf$l_sig_args-4(r2)[r1],stk_l_pc(fp);set original pc/psl
        04 AD   6241    D0    0710  1064          movl    chf$l_sig_args(r2)[r1],stk_l_psl(fp);
                   53    DD    0715  1065          pushl   r3
```

```
                                                D 1
                     51    02    C2  0717  1066          subl    #2,r1
                     53    51    D0  071A  1067          movl    r1,r3
                        6241    DD  071D  1068 25$:       pushl   chf$l_sig_args(r2)[r1]
                          FA 51    F5  0720  1069          sobgtr  r1,25$
          50    04 A2    03    00    EF  0723  1070          extzv   #0,#3,chf$l_sig_name(r2),r0
             6E    03    00    50    F0  0729  1071          insv    r0,#0,#3,(sp)
                              50    7C  072E  1072          clrq    r0
                00000000'GF    53    FB  0730  1073          calls   r3,g^lib$signal
                              53 8ED0  0737  1074          popl    r3
                           04 AD 8ED0  073A  1075          popl    stk_l_psl(fp)
                           10 AD 8ED0  073E  1076          popl    stk_l_pc(fp)
                        50    00'8F    9A  0742  1077          movzbl  #ss$_continue,r0
                           51    62    D0  0746  1078          movl    chf$l_sig_args(r2),r1
             FC A241    00000752'EF    9E  0749  1079          movab   40$,chf$l_sig_args-4(r2)[r1]
                                    04  0752  1080 40$:       ret
                                        0753  1081          .end
```

```
PLI
Sym

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


PSE
---

$AB
PLI
_PL


Pha
---
Ini
Com
Pas
Sym
Pas
```

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

```
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                       = 000000C0
```

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
_255DUA28:[PLIRTL.OBJ]PLIRTMAC.MLB;1	5
_255DUA28:[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
```

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

PLIDELETE
LIS

PLICONVRT
LIS

PLIDATA
LIS

PLICONTRL
LIS

PLIDATE
LIS

PLICVTPIC
LIS

PLIENVIR
LIS