I 4

```
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
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
```

```
**FILE**ID**PLIRECOPT
```

```
PPPPPPPPP   LL            IIIIII    RRRRRRRRR   EEEEEEEEEE    CCCCCCCC    OOOOOO    PPPPPPPPP   TTTTTTTTTT
PPPPPPPPP   LL            IIIIII    RRRRRRRRR   EEEEEEEEEE    CCCCCCCC    OOOOOO    PPPPPPPPP   TTTTTTTTTT
PP      PP  LL              II      RR      RR  EE          CC          OO      OO  PP      PP      TT
PP      PP  LL              II      RR      RR  EE          CC          OO      OO  PP      PP      TT
PP      PP  LL              II      RR      RR  EE          CC          OO      OO  PP      PP      TT
PP      PP  LL              II      RR      RR  EE          CC          OO      OO  PP      PP      TT
PPPPPPPPP   LL              II      RRRRRRRRR   EEEEEEEE    CC          OO      OO  PPPPPPPPP       TT
PPPPPPPPP   LL              II      RRRRRRRRR   EEEEEEEE    CC          OO      OO  PPPPPPPPP       TT
PP          LL              II      RR  RR      EE          CC          OO      OO  PP              TT
PP          LL              II      RR  RR      EE          CC          OO      OO  PP              TT
PP          LL              II      RR    RR    EE          CC          OO      OO  PP              TT          ....
PP          LL              II      RR    RR    EE          CC          OO      OO  PP              TT          ....
PP          LLLLLLLLLL    IIIIII    RR      RR  EEEEEEEEEE    CCCCCCCC    OOOOOO    PP              TT          ....
PP          LLLLLLLLLL    IIIIII    RR      RR  EEEEEEEEEE    CCCCCCCC    OOOOOO    PP              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
```

```
PL
Sy
AT
AT
AT
DA
DA
DA
DA
FA
FA
FA
FA
FC
FC
FC
FC
FC
FC
FC
FC
FC
FC
FC
FC
FC
FC
FC
FC
FC
FC
FC
FC
FC
FC
FC
FC
FC
FC
FC
FC
FC
FC
FC
KC
KC
KC
KC
KC
KC
KC
KC
KC
KC
KC
KC
KC
KC
KC
```

```
0000            .title  pli$recopt - pl1 runtime record io option processing
0000            .ident  /1-003/                                  ; Edit WHM1003
0000
0000    ;
0000    ;*****************************************************************************
0000    ;*                                                                           *
0000    ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000    ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000    ;*  ALL RIGHTS RESERVED.                                                     *
0000    ;*                                                                           *
0000    ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
0000    ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    ;*  TRANSFERRED.                                                             *
0000    ;*                                                                           *
0000    ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
0000    ;*  CORPORATION.                                                             *
0000    ;*                                                                           *
0000    ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    ;*                                                                           *
0000    ;*                                                                           *
0000    ;*****************************************************************************
0000    ;
0000
0000
0000    ;++
0000    ; facility:
0000    ;
0000    ;       VAX/VMS PL1 runtime library.
0000    ; abstract:
0000    ;
0000    ;       This module contains the pl1 runtime routines for processing the
0000    ;       common options for record i/o statements.
0000    ;
0000    ; author: c. spitz 7-feb-80
0000    ;
0000    ; modified:
0000    ;
0000    ;       Bill Matthews 22-Sep-81
0000    ;
0000    ;               Fix so that routine pli$$recidfrom does return with the
0000    ;               Z condition code bit clear if an rfa is specified.
0000    ;
0000    ;
0000    ;       1-003   Bill Matthews   29-September-1982
0000    ;
0000    ;               Invoke macros $defdat and rtshare instead of $defopr and share.
0000    ;
0000    ;--
0000
0000
0000    ;
0000    ; external definitions
```

```
PL
Sy
ST
STI
STI
STI
STI
STI
STI
STI
STI

PSI
---
.
$A
_P

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

Th
36
Th
60
16

Ma
--
_$
_$
TO

51

Th

MA
```

```
0000  ;
0000          $deffcb                         ;define file control block offsets
0000          $defstk                         ;define runtime stack
0000          $defdat                         ;define operand data types
0000          $defkcb                         ;define key control block offsets
0000          $fabdef                         ;define fab offsets
0000          $rabdef                         ;define rab offsets
0000
0000  ;
0000  ; local data
0000  ;
0000
0000          rtshare                         ;sharable
0000
0000
0000
```

```
                                 0000   ;++
                                 0000   ;pli$$keyto
                                 0000   ; this routine uses the current key number to determine the data type of
                                 0000   ; the key of reference and its position in the record. it allocates space
                                 0000   ; on the stack for a temporary, if the key is segmented, and 'unsegments'
                                 0000   ; the key. finally, the key is converted to the keyto destination.
                                 0000   ;               r0 - addr of keyto descr
                                 0000   ;                       0(r0) - addr of keyto
                                 0000   ;                       4(r0) - size/prec of keyto
                                 0000   ;                       8(r0) - data type of keyto
                                 0000   ;               r2 - addr of fcb
                                 0000   ;               r3 - fcb_l_attr
                                 0000   ;               r4 - addr of rab
                                 0000   ; outputs:
                                 0000   ;               r5,r6,r7,r8 - destroyed
                                 0000   ;--
                                 0000
                                 0000           .enabl  lsb
                                 0000   pli$$keyto_r8::
              56    6E    D0     0000           movl    (sp),r6                 ;save return addr
        77 00BC C2    05    E0     0003           bbs     #fab$v_bio,<fcb_b_fab+fab$b_fac>(r2),140$ ;if blockio, cont
           68 53    10    E1     0009           bbc     #atr_v_indexed,r3,110$  ;if indexed,
           55    35 A4    9A     000D           movzbl  rab$b_krf(r4),r5        ;get current key of ref
              55    2C    C4     0011           mull    #kcb_c_len,r5           ;get index to correct kcb
           55    38 A2    C0     0014           addl    fcb_l_kcb(r2),r5        ;
              65    0A    D1     0018           cmpl    #dat_k_char,kcb_l_dtyp(r5) ;character key?
                    2B    12     001B           bneq    60$                     ;if neq, no
    5E    000000FC 8F    C2     001D           subl    #252,sp                 ;get room for largest key possible
                    5E    DD     0024           pushl   sp                      ;set addr of temp
                    3D    BB     0026           pushr   #^m<r0,r2,r3,r4,r5>     ;save regs
           53    18 AE    9E     0028           movab   24(sp),r3               ;get start addr of temp
           57    24 A4    D0     002C           movl    rab$l_ubf(r4),r7        ;get addr of buffer
           58    08 A5    9E     0030           movab   kcb_w_pos0(r5),r8       ;get addr of pos0 in kcb
              51    88    3C     0034   10$:    movzwl  (r8)+,r1                ;get position in record
              51    57    C0     0037           addl    r7,r1                   ;add in addr of buffer
                    68    B5     003A           tstw    (r8)                    ;any thing left?
                    06    13     003C           beql    20$                     ;if eql, no
        63    61    88    28     003E           movc3   (r8)+,(r1),(r3)         ;copy this segment
                    F0    11     0042           brb     10$                     ;go again
                    3D    BA     0044   20$:    popr    #^m<r0,r2,r3,r4,r5>     ;restore regs
                    08    11     0046           brb     100$                    ;cont
           7E    08 A5    3C     0048   60$:    movzwl  kcb_w_pos0(r5),-(sp)    ;push position in buffer of key
           6E    24 A4    C0     004C           addl    rab$l_ubf(r4),(sp)      ;add in buffer addr
                    00    DD     0050   100$:   pushl   #0                      ;set dst offset
                 04 A0    DD     0052           pushl   4(r0)                   ;set dst prec
                 08 A0    DD     0055           pushl   8(r0)                   ;set dst data type
                    50    DD     0058           pushl   r0                      ;set addr of addr of dst
                    00    DD     005A           pushl   #0                      ;set src offset
        18 00BC C2    05    E0     005C           bbs     #fab$v_bio,<fcb_b_fab+fab$b_fac>(r2),130$ ;if blockio, cont
           14 53    10    E1     0062           bbc     #atr_v_indexed,r3,130$  ;if indexed
                 04 A5    DD     0066           pushl   kcb_l_prec(r5)          ;set src prec
                    65    DD     0069           pushl   kcb_l_dtyp(r5)          ;set src data type
                 1C AE    9F     006B   120$:   pushab  28(sp)                  ;set addr of addr of src
                    08    DD     006E           pushl   #8                      ;set number of args
                  00CD    30     0070           bsbw    key_cvrt                ;convert key into keyto, checking error
                    66    17     0073   sigerr: jmp     (r6)                    ;return
                 38 A4    9F     0075   110$:   pushab  rab$l_bkt(r4)           ;set addr of key
```

```
                        D6  11  0078      brb     100$                      ;cont
                        1F  DD  007A 130$: pushl   #31                       ;set prec of src
                        02  DD  007C      pushl   #dat_k_fix_bin            ;set data type of src
                        EB  11  007E      brb     120$                      ;cont
                     10 A4  9F  0080 140$: pushab  rab$w_rfa(r4)             ;set addr of key
                        CB  11  0083      brb     100$                      ;cont
                            0085          .dsabl  lsb
                            0085
                            0085     ;++
                            0085     ; pli$$key_hnd
                            0085     ; this routine is the condition handler for pli$$readkey and pli$writekey.
                            0085     ; it is jsb'd to from an entry mask that lies within the file control
                            0085     ; block. This vectoring is necessary, in order to easily get the address of the
                            0085     ; file control block.  After calculating the address of the file control
                            0085     ; block, this routine sets the error code in fab$l_error.
                            C085     ;
                            0085     ; inputs:
                            0085     ;       0(sp) - address of byte following jsb instruction in fcb.
                            0085     ; outputs:
                            0085     ;       none
                            0085     ;--
                            0085
                            0085     pli$$key_hnd::
    50    6E   000001B6 8F  C3  0085      subl3   #fcb_l_cndaddr+4,(sp),r0 ;get fcb address
          6E    01AE C0  9E  008D      movab   fcb_l_condit(r0),(sp)    ;get addr of handler
                  50  5D  D0  0092      movl    fp,r0                    ;start with this frame
                  52  50  D0  0095 10$:  movl    r0,r2                    ;save stack frame
               50  0C A0  D0  0098      movl    stk_l_fp(r0),r0          ;address next frame
            60   10  03  0D  009C      probew  #3,#stk_l_pc,(r0)        ;frame accessible?
                     18  13  00A0      beql    30$                      ;if eql then end of stack found, return
                  60  6E  D1  00A2      cmpl    (sp),stk_l_cnd_hnd(r0)   ;this frame's handler us?
                      EE  12  00A5      bneq    10$                      ;if neq, no, look at next frame
                      60  D4  00A7      clrl    stk_l_cnd_hnd(r0)        ;we aren't needed anymore
                  51  50  D0  00A9      movl    r0,r1                    ;set addr of frame to return to
               50  10 A2  D0  00AC      movl    stk_l_pc(r2),r0          ;set addr of pc to return to
               08 A2  5E  D0  00B0      movl    sp,stk_l_ap(r2)          ;set non-zero for status
              00000000'GF  17  00B4      jmp     g^pli$goto               ;return
    50        00000000'8F  D0  00BA 30$:  movl    #ss$_resignal,r0         ;set to resignal
                          04  00C1      ret                              ;return
                            00C2
                            00C2     ;++
                            00C2     ;pli$$readkey
                            00C2     ; this routine uses the current key number to determine the data type of
                            00C2     ; the key of reference. it then allocates space on the stack to store a
                            00C2     ; temporary of this data type. it converts the source key to the temporary
                            00C2     ; and stores the key address and size in the rab.
                            00C2     ; inputs:
                            00C2     ;       r0 - addr of key descr
                            00C2     ;               0(r0) - addr of key
                            00C2     ;               4(r0) - size/prec of key
                            00C2     ;               8(r0) - data type of key
                            00C2     ;       r2 - addr of fcb
                            00C2     ;       r3 - fcb_l_attr
                            00C2     ;       r4 - addr of rab
                            00C2     ; outputs:
                            00C2     ;       r6 is destroyed
                            00C2     ;--
```

```
                         00C2
                         00C2      pli$$readkey_r6::
           56    6E   D0 00C2                movl     (sp),r6                     ;save return addr
        3C 53    10   E1 00C5                bbc      #atr_v_indexed,r3,100$      ;if indexed,
        51    35 A4   9A 00C9                movzbl   rab$b_krf(r4),r1            ;get current key of ref
        3C A2    51   91 00CD                cmpb     r1,fcb_b_numkcbs(r2)        ;legal key number?
                 0A   1B 00D1                blequ    10$                         ;if lequ, yes
     50    00000000'8F D0 00D3               movl     #pli$_invindnum,r0          ;set invalid key number
               0378   31 00DA                brw      fail                        ;and fail
              51  2C  C4 00DD      10$:      mull     #kcb_c_len,r1               ;get index to correct kcb
        51    38 A2   C0 00E0                addl     fcb_l_kcb(r2),r1            ;
     5E    000000FC 8F C2 00E4               subl     #252,sp                     ;get room for largest key possible
                 5E   DD 00EB                pushl    sp                          ;set addr of dst
                 00   DD 00ED                pushl    #0                          ;set dst offset to zero
              04 A1   DD 00EF                pushl    kcb_l_prec(r1)              ;set dst prec
                 61   DD 00F2                pushl    kcb_l_dtyp(r1)              ;set dst data type
      34 A4   0A A1   90 00F4                movb     kcb_w_len0(r1),rab$b_ksz(r4) ;set size of key in rab if not char
              61  0A  D1 00F9                cmpl     #dat_k_char,kcb_l_dtyp(r1) ;character key?
                 13   12 00FC                bneq     110$                        ;if neq, no
      34 A4   04 A1   90 00FE                movb     kcb_l_prec(r1),rab$b_ksz(r4) ;set size of key in rab
                 0C   11 0103                brb      110$                        ;cont in common
                 5E   DD 0105      100$:     pushl    sp                          ;set addr of dst
                 00   DD 0107                pushl    #0                          ;set dst offset to zero
                 1F   DD 0109                pushl    #31                         ;set dst prec
                 02   DD 010B                pushl    #dat_k_fix_bin              ;set dst data type
         34 A4   04   90 010D                movb     #4,rab$b_ksz(r4)            ;set size of key in rab
              0C AE   9F 0111      110$:     pushab   12(sp)                      ;set dst addr
                 00   DD 0114                pushl    #0                          ;set src offset
              04 A0   DD 0116                pushl    4(r0)                       ;set src prec
              08 A0   DD 0119                pushl    8(r0)                       ;set src data type
                 50   DD 011C                pushl    r0                          ;set addr of addr of src
              53  50  D0 011E                movl     r0,r3                       ;copy key descr addr of onkey
                 08   DD 0121                pushl    #8                          ;set number of args
               001A   30 0123                bsbw     key_cvrt                    ;convert src to key data type
         5E   28 AE   9E 0126                movab    40(sp),sp                   ;clean stack
     0A 00BC C2   05  E0 012A                bbs      #fab$v_bio,<fcb_b_fab+fab$b_fac>(r2),130$ ;if blockio, cont
         30 A4   5E   D0 0130                movl     sp,rab$l_kbf(r4)            ;set key buffer addr in rab
         1E A4   01   90 0134                movb     #rab$c_key,rab$b_rac(r4) ;set keyed access in rab
                 66   17 0138                jmp      (r6)                        ;return
         38 A4  8ED0     013A      130$:     popl     rab$l_bkt(r4)               ;set virt block num for block io
                 66   17 013E                jmp      (r6)                        ;return;
                         0140
                         0140      key_cvrt:
                 5C   DD 0140                pushl    ap                          ;save ap
                 5C   D4 0142                clrl     ap                          ;set up status temp for handler
      6D    01AE C2   9E 0144                movab    fcb_l_condit(r2),(fp)       ;set up condition handler
 00000000'GF   08 AE  FA 0149                callg    8(sp),g^pli$cvrt_any        ;do the conversion
                 6D   D4 0151                clrl     (fp)                        ;we don't need handler anymore
                 5C   D5 0153                tstl     ap                          ;conversion fail?
                 04   12 0155                bneq     10$                         ;if neq, yes, do onkey
              5C 8ED0    0157                popl     ap                          ;restore ap
                 05      015A                rsb                                  ;return
        50   FF14 CF  9E 015B      10$:      movab    sigerr,r0                   ;get return addr for keyto conversion
         50   04 AE   D1 0160                cmpl     4(sp),r0                    ;should we signal error?
                 0A   12 0164                bneq     20$                         ;if neq, no, do onkey
     50    00000000'8F D0 0166     15$:      movl     #pli$_cnverr,r0             ;set conversion error
               02E5   31 016D                brw      fail                        ;and fail
```

```
             20 AE    0A   D1  0170  20$:    cmpl    #dat_k_char,32(sp)      ;were we converting to char?
                      F0   13  0174          beql    15$                     ;if eql, yes, it failed once, so no
                               0176                                          ;onkey value will be returned
             20 AE    0B   D0  0176          movl    #dat_k_char_var,32(sp)  ;set for char var dest
      24 AE   00000100 8F  D0  017A          movl    #256,36(sp)             ;set for length of 256
      5E      00000102 8F  C2  0182          subl    #258,sp                 ;get room for char temp
           012E CE    5E   D0  0189          movl    sp,258+44(sp)           ;set addr of dest
 00000000'GF    010A CE    FA  018E          callg   258+8(sp),g^pli$cvrt_any ;convert it to char var
                      5E   DD  0197          pushl   sp                      ;set addr of onkey value
                      52   DD  0199          pushl   r2                      ;set fcb address
              00000000'8F  DD  019B          pushl   #pli$_cnverr            ;set conversion error
              00000000'8F  DD  01A1          pushl   #pli$_key               ;set key condition
     00000000'GF      04   FB  01A7          calls   #4,g^pli$io_error       ;signal condition
                           04  01AE          ret                             ;return
                               01AF
                               01AF  ;++
                               01AF  ;pli$$chk_keycnd
                               01AF  ; this routine checks the rms rab for status that should be signaled as
                               01AF  ; the pl1 key condition. if such a status is found, the onkey value is
                               01AF  ; calculated, and io_error is called to signal the key condition.
                               01AF  ; inputs:
                               01AF  ;       r0 = error sub-code
                               01AF  ;       r2 = fcb address
                               01AF  ;       r3 = address of key descriptor
                               01AF  ;       r4 = rab address
                               01AF  ; outputs:
                               01AF  ;       none
                               01AF  ;--
                               01AF
                               01AF  pli$$chk_keycnd::
      50      00000000'8F  D1  01AF          cmpl    #pli$_rmsr,r0           ;rab error?
                      01   13  01B6          beql    20$                     ;if eql, yes, cont
                      05       01B8  10$:    rsb                             ;return
                      53   D5  01B9  20$:    tstl    r3                      ;key specified?
                      FB   13  01BB          beql    10$                     ;if eql, no, just return
                      63   D5  01BD          tstl    (r3)                    ;key addr specified?
                      F7   13  01BF          beql    10$                     ;if eql, no, just return
    08 A4     00000000'8F  D1  01C1          cmpl    #rms$_rnf,rab$l_sts(r4) ;record not found?
                      28   13  01C9          beql    30$                     ;if eql, yes, raise key
    08 A4     00000000'8F  D1  01CB          cmpl    #rms$_key,rab$l_sts(r4) ;key error?
                      1E   13  01D3          beql    30$                     ;if eql, yes, raise key
    08 A4     00000000'8F  D1  01D5          cmpl    #rms$_mrn,rab$l_sts(r4) ;max rec num exceeded?
                      14   13  01DD          beql    30$                     ;if eql, yes, raise key
    08 A4     00000000'8F  D1  01DF          cmpl    #rms$_dup,rab$l_sts(r4) ;duplicate key?
                      0A   13  01E7          beql    30$                     ;if eql, yes, raise key
    08 A4     00000000'8F  D1  01E9          cmpl    #rms$_rex,rab$l_sts(r4) ;record already exists?
                      C5   12  01F1          bneq    10$                     ;if neq, no, just return
      5E      00000102 8F  C2  01F3  30$:    subl    #258,sp                 ;get room for vcha temp
                      5E   DD  01FA          pushl   sp                      ;set addr of dest
                      52   DD  01FC          pushl   r2                      ;set fcb addr
                      50   DD  01FE          pushl   r0                      ;set error code
              00000000'8F  DD  0200          pushl   #pli$_key               ;set key condition
                      00   DD  0206          pushl   #0                      ;set dest offset
              00000100 8F  DD  0208          pushl   #256                    ;set dest size
                      0B   DD  020E          pushl   #dat_k_char_var         ;set dest data type
                 18 AE     9F  0210          pushab  24(sp)                  ;set addr of addr of dest
                      00   DD  0213          pushl   #0                      ;set src offset
```

```
            04 A3   DD  0215   303          pushl   4(r3)                   ;set src prec
            08 A3   DD  0218   304          pushl   8(r3)                   ;set src data type
               53   DD  021B   305          pushl   r3                      ;set addr of addr of src
 00000000'GF   08   FB  021D   306          calls   #8,g^pli$cvrt_any       ;convert key to vcha
 00000000'GF   04   FB  0224   307          calls   #4,g^pli$io_error       ;signal the condition
                    04  022B   308          ret                             ;return
                        022C   309
                        022C   310 ;++
                        022C   311 ;pli$$writekey
                        022C   312 ; this routine uses the current key number to determine the data type of
                        022C   313 ; the key of reference and its position within the record if the file is
                        022C   314 ; indexed.  it allocates space on the stack to store a temporary for relative
                        022C   315 ; file keys. it converts the source key to the temporary (or directly into
                        022C   316 ; the record to be written) and stores the key address and size in the rab.
                        022C   317 ; the key number option must have been processed as well as the from reference
                        022C   318 ; before this routine is called
                        022C   319 ; inputs:
                        022C   320 ;        r0 - addr of key descr
                        022C   321 ;                0(r0) - addr of key
                        022C   322 ;                4(r0) - size/prec of key
                        022C   323 ;                8(r0) - data type of key
                        022C   324 ;        r2 - addr of fcb
                        022C   325 ;        r3 - fcb_l_attr
                        022C   326 ;        r4 - addr of rab
                        022C   327 ; outputs:
                        022C   328 ;        r3,r5,r6,r7,r8 - destroyed
                        022C   329 ;--
                        022C   330
                        022C   331 pli$$writekey_r8::
          56    6E  D0  022C   332          movl    (sp),r6                 ;save return addr
   31 00BC C2   05  E0  022F   333          bbs     #fab$v_bio,<fcb_b_fab+fab$b_fac>(r2),100$ ;if blockio, cont
       2D 53    10  E1  0235   334          bbc     #atr_v_indexed,r3,100$  ;if indexed,
      55    35 A4   9A  0239   335          movzbl  rab$b_krf(r4),r5        ;get current key of ref
       3C A2    55  91  023D   336          cmpb    r5,fcb_b_numkcbs(r2)    ;legal key number?
                0A  1B  0241   337          blequ   10$                     ;if lequ, yes
 50   00000000'8F   D0  0243   338          movl    #pli$_invindnum,r0      ;set invalid key number
              0208  31  024A   339          brw     fail                    ;and fail
          55    2C  C4  024D   340 10$:     mull    #kcb_c_len,r5           ;get index to correct kcb
      55    38 A2   C0  0250   341          addl    fcb_l_kcb(r2),r5        ;
 5E    000000FC 8F  C2  0254   342          subl    #252,sp                 ;get room for largest key possible
                5E  DD  025B   343          pushl   sp                      ;set addr of dst
                00  DD  025D   344          pushl   #0                      ;set dst offset to zero
            04 A5   DD  025F   345          pushl   kcb_l_prec(r5)          ;set dst prec
                65  DD  0262   346          pushl   kcb_l_dtyp(r5)          ;set dst data type
                08  11  0264   347          brb     110$                    ;cont in common
                5E  DD  0266   348 100$:    pushl   sp                      ;set addr of dst
                00  DD  0268   349          pushl   #0                      ;set dst offset to zero
                1F  DD  026A   350          pushl   #31                     ;set dst prec
                02  DD  026C   351          pushl   #dat_k_fix_bin          ;set dst data type
            0C AE   9F  026E   352 110$:    pushab  12(sp)                  ;set dst addr
                00  DD  0271   353          pushl   #0                      ;set src offset
            04 A0   DD  0273   354          pushl   4(r0)                   ;set src prec
            08 A0   DD  0276   355          pushl   8(r0)                   ;set src data type
                50  DD  0279   356          pushl   r0                      ;set addr of addr of src
          53    50  D0  027B   357          movl    r0,r3                   ;copy key descr addr for onkey
                08  DD  027E   358          pushl   #8                      ;set number of args
              FEBD  30  0280   359          bsbw    key_cvrt                ;convert src to key data type
            5E    28 AE    9E  0283             movab   40(sp),sp                     ;clean stack
      59 00BC C2     05    E0  0287             bbs     #fab$v_bio,<fcb_b_fab+fab$b_fac>(r2),300$ ;if blockio, cont
          46 0C A2   10    E1  028D             bbc     #atr_v_indexed,fcb_l_attr(r2),200$ ;if indexed
                65   0A    D1  0292             cmpl    #dat_k_char,kcb_l_dtyp(r5) ;character key?
                     28    12  0295             bneq    160$                          ;if neq, no
                     1C    BB  0297             pushr   #^m<r2,r3,r4>                 ;save regs
            51    0C AE    9E  0299             movab   12(sp),r1                     ;get start addr of key
            50    04 A5    D0  029D             movl    kcb_l_prec(r5),r0             ;get total length of key
            34 A4    50    90  02A1             movb    r0,rab$b_ksz(r4)              ;set size in rab
            57    28 A4    D0  02A5             movl    rab$l_rbf(r4),r7              ;get addr of buffer
            58    08 A5    9E  02A9             movab   kcb_w_pos0(r5),-8             ;get addr of pos0 in kcb
                53   88    3C  02AD     120$:   movzwl  (r8)+,r3                      ;get position in record
                53   57    C0  02B0             addl    r7,r3                         ;add addr of buffer
  63    88    20    61   50    2C  02B3             movc5   r0,(r1),#^x20,(r8)+,(r3) ;copy this segment
                     F2    1A  02B9             bgtru   120$                          ;if gtru, more to do, go again
                     1C    BA  02BB             popr    #^m<r2,r3,r4>                 ;restore regs
                     1D    11  02BD             brb     210$                          ;cont
            51    08 A5    3C  02BF     160$:   movzwl  kcb_w_pos0(r5),r1             ;get position in buffer
            51    28 A4    C0  02C3             addl    rab$l_rbf(r4),r1              ;add in addr of buffer
                     1C    BB  02C7             pushr   #^m<r2,r3,r4>                 ;save regs
         34 A4    0A A5    90  02C9             movb    kcb_w_len0(r5),rab$b_ksz(r4) ;set size of key in rab
    61    0C AE    0A A5    28  02CE             movc3   kcb_w_len0(r5),12(sp),(r1) ;copy key to buffer
                     1C    BA  02D4             popr    #^m<r2,r3,r4>                 ;restore regs
                     04    11  02D6             brb     210$                          ;cont
            34 A4    04    90  02D8     200$:   movb    #4,rab$b_ksz(r4)              ;set size of key in rab
            30 A4    5E    D0  02DC     210$:   movl    sp,rab$l_kbf(r4)              ;set key buffer addr in rab
            1E A4    01    90  02E0             movb    #rab$c_key,rab$b_rac(r4) ;set keyed access in rab
                66   17        02E4             jmp     (r6)                          ;return
          38 A4 8ED0           02E6     300$:   popl    rab$l_bkt(r4)                 ;set lbn for block io
                66   17        02EA             jmp     (r6)                          ;return
                               02EC             .enabl  lsb
                               02EC
                               02EC     ;++
                               02EC     ;pli$$keynum
                               02EC     ; inputs
                               02EC     ;               r0 - addr of keynum option
                               02EC     ;               r1 - addr of key option
                               02EC     ;               r2 - addr of fcb
                               02EC     ;               r3 - fcb_l_attr
                               02EC     ;               r4 - addr of rab
                               02EC     ; outputs
                               02EC     ;               r1 - addr of key option
                               02EC     ;--
                               02EC
                               02EC     pli$$keynum::
                     50    D5  02EC             tstl    r0                            ;keynum passed?
                     56    13  02EE             beql    90$                           ;if eql, no, just return
             0A 53   10    E0  02F0             bbs     #atr_v_indexed,r3,20$         ;if indexed, cont
      50    00000000'8F    D0  02F4     10$:    movl    #pli$_notindexed,r0           ;set not indexed file
                   0157    31  02FB             brw     fail                          ;and fail
             0A 53   08    E0  02FE     20$:    bbs     #atr_v_keyed,r3,40$           ;if keyed, cont
      50    00000000'8F    D0  0302     30$:    movl    #pli$_notkeyd,r0              ;set not keyed file
                   0149    31  0309             brw     fail                          ;and fail
       0A 00BC C2    05    E1  030C     40$:    bbc     #fab$v_bio,<fcb_b_fab+fab$b_fac>(r2),45$ ;if not blockio, cont
      50    00000000'8F    D0  0312     42$:    movl    #pli$_conblokio,r0            ;set conflicting block io
                   0139    31  0319             brw     fail                          ;and fail
                     51    D5  031C     45$:    tstl    r1                            ;key specified?
```

```
                    0A  12  031E         bneq    60$                          ;if neq, yes, cont
    50   00000000'8F    D0  0320  50$:   movl    #pli$_nokey,r0               ;set no key specified
                  012B  31  0327         brw     fail                         ;and fail
              50    60  3C  032A  60$:   movzwl  (r0),r0                      ;get key number
                    0A  18  032D         bgeq    80$                          ;if geq, cont
    50   00000000'8F    D0  032F  70$:   movl    #pli$_invindnum,r0           ;set invalid key number
                  011C  31  0336         brw     fail                         ;and fail
         7E   3C    A2  9A  0339  80$:   movzbl  fcb_b_numkcbs(r2),-(sp) ;get highest key number + 1
              8E    50  D1  033D         cmpl    r0,(sp)+                     ;key num too big?
                    ED  18  0340         bgeq    70$                          ;if geq, then yes, fail
         35 A4      50  90  0342         movb    r0,rab$b_krf(r4)             ;set key of ref in rab
                        05  0346  90$:   rsb
                            0347
                            0347  ;++
                            0347  ;pli$$matchgtr
                            0347  ; inputs:
                            0347  ;         r0 - addr of match_greater
                            0347  ;         r1 - addr of key
                            0347  ;         r2 - addr of fcb
                            0347  ;         r3 - fcb_l_attr
                            0347  ;         r4 - addr of rab
                            0347  ; outputs:
                            0347  ;         r1 - addr of key
                            0347  ;--
                            0347
                            0347  pli$$matchgtr::
                    16  DD  0347         pushl   #rab$v_kgt                   ;set field index of key greater
                    02  11  0349         brb     100$                         ;cont in common
                            034B
                            034B  ;++
                            034B  ;pli$$matchgeq
                            034B  ; inputs:
                            034B  ;         r0 - addr of match_greater_eql
                            034B  ;         r1 - addr of key
                            034B  ;         r2 - addr of fcb
                            034B  ;         r3 - fcb_l_attr
                            034B  ;         r4 - addr of rab
                            034B  ; outputs
                            034B  ;         r1 - addr of key
                            034B  ;--
                            034B
                            034B  pli$$matchgeq::
                    15  DD  034B         pushl   #rab$v_kge                   ;set field index of key greater
                            034D  100$:
                    50  D5  034D         tstl    r0                           ;match gtr specified?
                    14  13  034F         beql    110$                         ;if eql, no, just return
         AD 53      08  E1  0351         bbc     #atr_v_keyed,r3,30$          ;if not keyed file, fail
    B7 00BC C2      05  E0  0355         bbs     #fab$v_bio,<fcb_b_fab+fab$b_fac>(r2),42$ ;if blockio, fail
                    51  D5  035B         tstl    r1                           ;key specified?
                    C1  13  035D         beql    50$                          ;if eql, no, fail
 04 A4    01    6E  60  F0  035F         insv    (r0),(sp),#1,rab$l_rop(r4) ;set match greater or greatereql
         5E    04   AE  9E  0365  110$:  movab   4(sp),sp                     ;remove index
                        05  0369         rsb                                  ;return
                            036A
                            036A  ;++
                            036A  ;pli$$valrecidto
                            036A  ; inputs:
```

```
                          036A    ;       r0 - addr of record id to
                          036A    ;       r2 - addr of fcb
                          036A    ;       r3 - fcb_l_attr
                          036A    ;       r4 - addr of rab
                          036A    ;--
                          036A
                          036A    pli$$valrecidto::
              50    D5    036A            tstl    r0                      ;record id to specified?
              0E    13    036C            beql    120$                    ;if eql, no, just return
        0A 53 0F    E0    036E            bbs     #atr_v_recidacc,r3,120$ ;if rec id, cont
  50   00000000'8F  D0    0372    115$:   movl    #pli$_recid,r0          ;set not record id file
            00D9    31    0379            brw     fail                    ;and fail
                    05    037C    120$:   rsb                             ;return
                          037D
                          037D    ;++
                          037D    ;pli$$recidfrom
                          037D    ; inputs:
                          037D    ;       r0 - addr of record id from
                          037D    ;       r1 - addr of key
                          037D    ;       r2 - addr of fcb
                          037D    ;       r3 - fcb_l_attr
                          037D    ;       r4 - addr of rab
                          037D    ; outputs:
                          037D    ;       condition code z bit is clear if rfa from specified
                          037D    ;--
                          037D
                          037D    pli$$recidfrom::
              50    D5    037D            tstl    r0                      ;record id from specified?
              FB    13    037F            beql    120$                    ;if eql, no, just return
        ED 53 0F    E1    0381            bbc     #atr_v_recidacc,r3,115$ ;if not recid, fail
              51    D5    0385            tstl    r1                      ;key also specified?
              0A    13    0387            beql    130$                    ;if eql, no, cont
  50   00000000'8F  D0    0389            movl    #pli$_recidkey,r0       ;set record id and key conflict
            00C2    31    0390            brw     fail                    ;and fail
        10 A4 60    7D    0393    130$:   movq    (r0),rab$w_rfa(r4)      ;set recid in rab
        1E A4 02    90    0397            movb    #rab$c_rfa,rab$b_rac(r4) ;set for rfa access in rab
           35 A4    94    039B            clrb    rab$b_krf(r4)           ;reset key of ref in rab
              04    B9    039E            bicpsw  #4                      ;clear condition code Z bit
                    05    03A0            rsb                             ;return
                          03A1
                          03A1    ;++
                          03A1    ;pli$$fxdctlfrom
                          03A1    ; inputs:
                          03A1    ;       r0 - addr of fixed control from descr. 0(r0) is the addr, 4(r0)
                          03A1    ;               is word size/prec, 6(r0) is word data type
                          03A1    ;       r2 - addr of fcb
                          03A1    ;       r3 - fcb_l_attr
                          03A1    ;       r4 - addr of rab
                          03A1    ;--
                          03A1
                          03A1    pli$$fxdctlfrom::
           2C A4    D4    03A1            clrl    rab$l_rhb(r4)           ;assume not specified
              60    DD    03A4            pushl   (r0)                    ;save addr of fixed ctl
              45    13    03A6            beql    150$                    ;if eql, none specified
 0A 00BC C2 05      E1    03A8            bbc     #fab$v_bio,<fcb_b_fab+fab$b_fac>(r2),134$ ;if not blockio, cont
  50   00000000'8F  D0    03AE    133$:   movl    #pli$_conblokio,r0      ;set conflicts with blockio
            009D    31    03B5            brw     fail                    ;and fail
```

```
PLI
Sym

$$.
$$.
ATR
ATR
ATR
ATR
ATR
ATR
ATR
ATR
ATR
ATR
ATR
ATR
ATR
ATR
ATR
ATR
ATR
DAT
DIR
FAB
FAB
FAI
FCB
FCB
FCB
FCB
FCB
FCB
FCB
FCB
FCB
FCB
FCB
FCB
FCB
FCB
FCB
FCB
FCB
FCB
FCB
FCB
FCB
FCB
FCB
FCB
FCB
FCB
FCB
FCB
FCB
FCB
FCB
FCB
FR
```

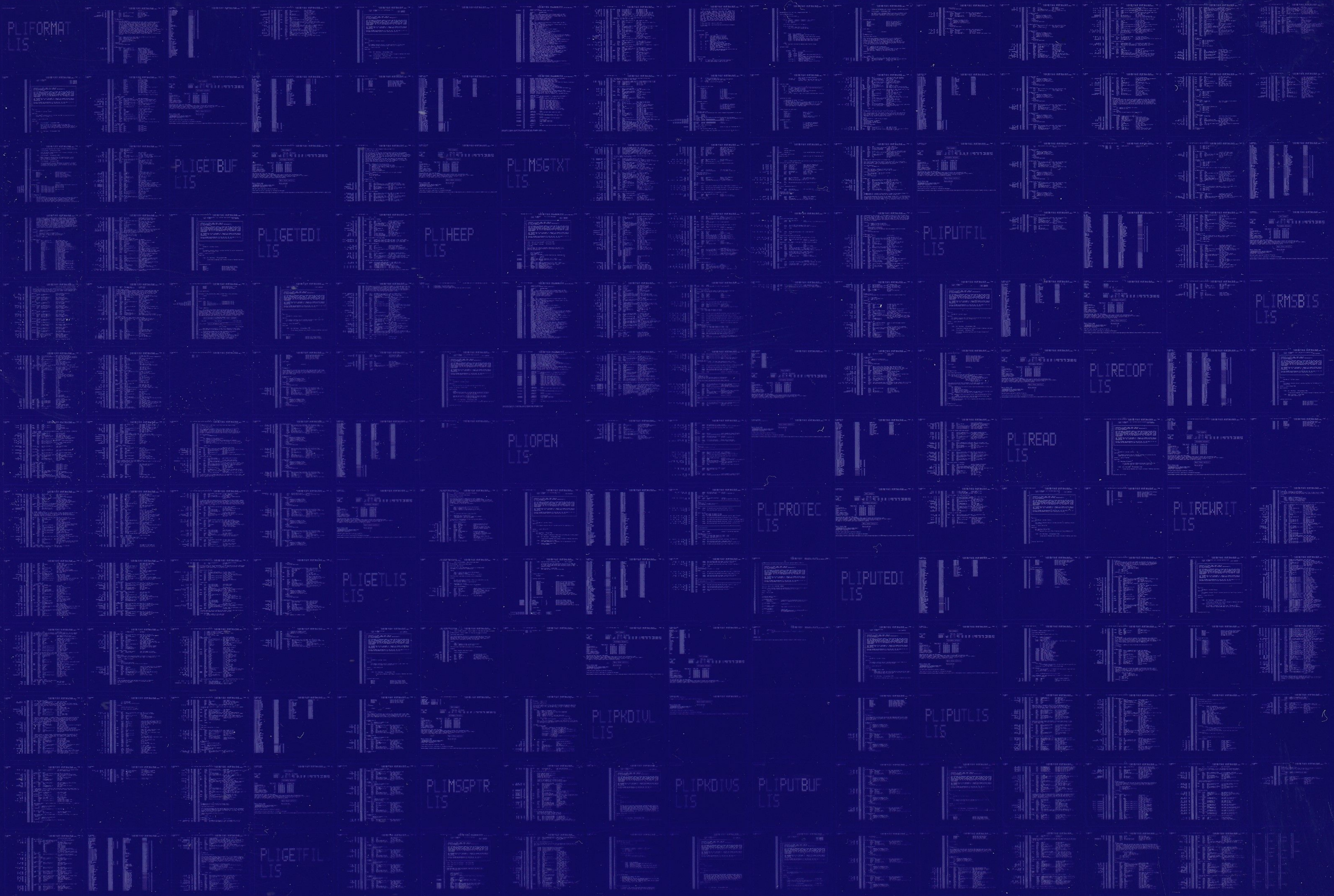
```
        06 A0     0B    B1  03B8    531 134$:   cmpw    #dat_k_char_var,6(r0)   ;is it char var?
                  06    12  03BC    532         bneq    135$                    ;if neq, no
       51    00 B0      3C  03BE    533         movzwl  @(r0),r1                ;get current size
                  0D    11  03C2    534         brb     137$                    ;cont
           04 A0        DD  03C4    535 135$:   pushl   4(r0)                   ;set size, datatyp for bysize
   00000000'GF    01    FB  03C7    536         calls   #1,g^pli$$bytesize      ;get size
           15 50        E9  03CE    537         blbc    r0,139$                 ;if lbc, fail
                  51    D5  03D1    538 137$:   tstl    r1                      ;anything there?
                  18    15  03D3    539         bleq    150$                    ;if leq, no, just return
      50    00E5 C2     9A  03D5    540         movzbl  <fcb_b_fab+fab$b_fsz>(r2),r0 ;get fixed control size
          51      50    D1  03DA    541         cmpl    r0,r1                   ;enuf room?
                  09    13  03DD    542         beql    140$                    ;if geq, yes, cont
   50    00000000'8F    D0  03DF    543         movl    #pli$_fxcsiz,r0         ;set fixed control wrong size
                  6F    11  03E6    544 139$:   brb     fail                    ;and fail
            2C    8ED0      03E8    545 140$:   popl    rab$l_rhb(r4)           ;set addr in rab
                  05        03EC    546         rsb                             ;return
       5E    04 AE      9E  03ED    547 150$:   movab   4(sp),sp                ;clean stack
                  05        03F1    548         rsb                             ;return
                            03F2    549
                            03F2    550 ;++
                            03F2    551 ;pli$$fxctlto_r6
                            03F2    552 ; inputs:
                            03F2    553 ;               r0 - addr of fixed control to descr. 0(r0) is addr, 4(r0) is
                            03F2    554 ;                       word size/prec, 6(r0) is word data type
                            03F2    555 ;               r2 - addr of fcb
                            03F2    556 ;               r3 - fcb_l_attr
                            03F2    557 ;               r4 - addr of rab
                            03F2    558 ; outputs
                            03F2    559 ;               r6 is destroyed
                            03F2    560 ;--
                            03F2    561
                            03F2    562 pli$$fxctlto_r6::
           2C A4        D4  03F2    563         clrl    rab$l_rhb(r4)           ;assume not specified
          56      50    D0  03F5    564         movl    r0,r6                   ;save addr of fixed ctl descr
                  60    D5  03F8    565         tstl    (r0)                    ;address specified?
                  58    13  03FA    566         beql    160$                    ;if eql, no, just return
   AC 00BC C2     05    E0  03FC    567         bbs     #fab$v_bio,<fcb_b_fab+fab$b_fac>(r2),133$ ;if blockio, fail
           04 A0        DD  0402    568         pushl   4(r0)                   ;set size, datatyp for bysize
   00000000'GF    01    FB  0405    569         calls   #1,g^pli$$bytesize      ;get size
           46 50        E9  040C    570         blbc    r0,fail                 ;if lbc, fail
                  51    D5  040F    571         tstl    r1                      ;anything there?
                  41    15  0411    572         bleq    160$                    ;if leq, no, just return
      50    00E5 C2     9A  0413    573         movzbl  <fcb_b_fab+fab$b_fsz>(r2),r0 ;get fixed control size
        06 A6     0B    B1  0418    574         cmpw    #dat_k_char_var,6(r6)   ;char var dest?
                  10    12  041C    575         bneq    170$                    ;if neq, no
          51      50    D1  041E    576         cmpl    r0,r1                   ;enuf room?
                  10    14  0421    577         bgtr    172$                    ;if gtr, no
          51      66    D0  0423    578         movl    (r6),r1                 ;get addr of dest
          81      50    B0  0426    579         movw    r0,(r1)+                ;set size to length of fixed control
        2C A4     51    D0  0429    580         movl    r1,rab$l_rhb(r4)        ;set addr in rab
                  05        042D    581         rsb                             ;return
          51      50    D1  042E    582 170$:   cmpl    r0,r1                   ;right size?
                  09    13  0431    583         beql    175$                    ;if eql, yes, cont
   50    00000000'8F    D0  0433    584 172$:   movl    #pli$_fxcsiz,r0         ;set fixed control wrong size
                  19    11  043A    585         brb     fail                    ;and fail
    06 A6    0057 8F    B1  043C    586 175$:   cmpw    #<dat_k_structure+64>,6(r6) ;bit sized structure?
                  0C    12  0442    587         bneq    176$                    ;if neq, no, cont
            50 8ED0  0444   588           popl    r0                        ;get ret addr
         5E    51  C2  0447   589           subl    r1,sp                     ;get size for temp on stack
      2C A4    5E  D0  044A   590           movl    sp,rab$l_rhb(r4)          ;set addr of temp
               60  17  044E   591           jmp     (r0)                      ;return
      2C A4    66  D0  0450   592 176$:     movl    (r6),rab$l_rhb(r4)        ;set addr in rab
                   05  0454   593 160$:     rsb                               ;return
                       0455   594
                       0455   595           .dsabl lsb
                       0455   596
               52  DD  0455   597 fail:     pushl   r2                        ;push fcb addr
               50  DD  0457   598           pushl   r0                        ;push error code
       00000000'8F  DD  0459   599           pushl   #pli$_error               ;push error condition
  00000000'GF    03  FB  045F   600           calls   #3,g^pli$io_error         ;signal the error
                   04  0466   601           ret                               ;
                       0467   602
                       0467   603           .end
```

```
ATR_V_INDEXED                 = 00000010                   KCB_W_POS3                     00000014
ATR_V_KEYED                   = 00000008                   KCB_W_POS4                     00000018
ATR_V_RECIDACC                = 0000000F                   KCB_W_POS5                     0000001C
DAT_K_CHAR                    = 0000000A                   KCB_W_POS6                     00000020
DAT_K_CHAR_VAR                = 0000000B                   KCB_W_POS7                     00000024
DAT_K_FIX_BIN                 = 00000002                   KEY_CVRT                       00000140 R      02
DAT_K_STRUCTURE               = 00000017                   PLI$$BYTESIZE                  ********    X   02
FAB$B_FAC                     = 00000016                   PLI$$CHK_KEYCND                000001AF RG     02
FAB$B_FSZ                     = 0000003F                   PLI$$FXCTLTO_R6                000003F2 RG     02
FAB$V_BIO                     = 00000005                   PLI$$FXDCTLFROM                000003A1 RG     02
FAIL                            00000455 R      02         PLI$$KEYNUM                    000002EC RG     02
FCB_B_ENVIR                     000001C2                   PLI$$KEYTO_R8                  00000000 RG     02
FCB_B_ESA                       0000012E                   PLI$$KEY_HND                   00000085 RG     02
FCB_B_EXTRA                     0000003D                   PLI$$MATCHGEQ                  0000034B RG     02
FCB_B_FAB                       000000A6                   PLI$$MATCHGTR                  00000347 RG     02
FCB_B_IDENT                     00000040                   PLI$$READKEY_R6                000000C2 RG     02
FCB_B_IDENT_NAM                 00000042                   PLI$$RECIDFROM                 0000037D RG     02
FCB_B_NAM                       000000F6                   PLI$$VALRECIDTO                0000036A RG     02
FCB_B_NUMKCBS                   0000003C                   PLI$$WRITEKEY_R8               0000022C RG     02
FCB_B_RAB                       00000062                   PLI$CVRT_ANY                   ********    X   02
FCB_C_LEN                       000001C2                   PLI$GOTO                       ********    X   02
FCB_C_STRLEN                    00000034                   PLI$IO_ERROR                   ********    X   02
FCB_L_ATTR                      0000000C                   PLI$_CNVERR                    ********    X   02
FCB_L_BUF                       00000014                   PLI$_CONBLOKIO                 ********    X   02
FCB_L_BUF_END                   00000018                   PLI$_ERROR                     ********    X   02
FCB_L_BUF_PT                    0000001C                   PLI$_FXCSIZ                    ********    X   02
FCB_L_CNDADDR                   000001B2                   PLI$_INVINDNUM                 ********    X   02
FCB_L_CONDIT                    000001AE                   PLI$_KEY                       ********    X   02
FCB_L_DTTR                      00000010                   PLI$_NOKEY                     ********    X   02
FCB_L_ERROR                     00000008                   PLI$_NOTINDEXFD                ********    X   02
FCB_L_KCB                       00000038                   PLI$_NOTKEYD                   ********    X   02
FCB_L_NEXT                      00000000                   PLI$_RECID                     ********    X   02
FCB_L_PREVIOUS                  00000004                   PLI$_RECIDKEY                  ********    X   02
FCB_L_PRN                       00000034                   PLI$_RMSR                      ********    X   02
FCB_Q_RFA                       00000020                   RAB$B_KRF                    = 00000035
FCB_W_COLUMN                    0000002E                   RAB$B_KSZ                    = 00000034
FCB_W_IDENT_LEN                 00000040                   RAB$B_RAC                    = 0000001E
FCB_W_LINE                      00000030                   RAB$C_KEY                    = 00000001
FCB_W_LINESIZE                  0000002A                   RAB$C_RFA                    = 00000002
FCB_W_PAGE                      00000032                   RAB$L_BKT                    = 00000038
FCB_W_PAGESIZE                  0000002C                   RAB$L_KBF                    = 00000030
FCB_W_REVISION                  00000028                   RAB$L_RBF                    = 00000028
KCB_C_LEN                       0000002C                   RAB$L_RHB                    = 0000002C
KCB_L_DTYP                      00000000                   RAB$L_ROP                    = 00000004
KCB_L_PREC                      00000004                   RAB$L_STS                    = 00000008
KCB_L_ZERO                      00000028                   RAB$L_UBF                    = 00000024
KCB_W_LEN0                      0000000A                   RAB$V_KGE                    = 00000015
KCB_W_LEN1                      0000000E                   RAB$V_KGT                    = 00000016
KCB_W_LEN2                      00000012                   RAB$W_RFA                    = 00000010
KCB_W_LEN3                      00000016                   RMS$_DUP                       ********    X   02
KCB_W_LEN4                      0000001A                   RMS$_KEY                       ********    X   02
KCB_W_LEN5                      0000001E                   RMS$_MRN                       ********    X   02
KCB_W_LEN6                      00000022                   RMS$_REX                       ********    X   02
KCB_W_LEN7                      00000026                   RMS$_RNF                       ********    X   02
KCB_W_POS0                      00000008                   SIGERR                         00000073 R      02
KCB_W_POS1                      0000000C                   SIZ...                       = 00000001
KCB_W_POS2                      00000010                   SS$_RESIGNAL                   ********    X   02
```

```
                                                              H 15
PLI$RECOPT                       - pl1 runtime record io option processin 16-SEP-1984 02:25:33  VAX/VMS Macro V04-00       Page  14
Symbol table                                                              6-SEP-1984 11:39:43  [PLIRTL.SRC]PLIRECOPT.MAR;1      (1)

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

                                                 +----------------+
                                                 ! 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                       00000467  ( 1127.)  02 (  2.)    PIC    USR    CON    REL    LCL   SHR   EXE   RD   NOWRT NOVEC LONG

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

Phase                     Page faults    CPU Time       Elapsed Time
-----                     -----------    --------       ------------
Initialization                     10    00:00:00.10    00:00:00.45
Command processing                 83    00:00:00.62    00:00:01.80
Pass 1                            184    00:00:06.52    00:00:13.46
Symbol table sort                   0    00:00:00.64    00:00:01.02
Pass 2                            110    00:00:01.72    00:00:03.87
Symbol table output                15    00:00:00.10    00:00:00.10
Psect synopsis output               2    00:00:00.02    00:00:00.02
Cross-reference output              0    00:00:00.00    00:00:00.00
Assembler run totals              404    00:00:09.72    00:00:20.72

The working set limit was 1200 pages.
36133 bytes (71 pages) of virtual memory were used to buffer the intermediate code.
There were 30 pages of symbol table space allocated to hold 506 non-local and 56 local symbols.
603 source lines were read in Pass 1, producing 14 object records in Pass 2.
16 pages of virtual memory were used to define 14 macros.

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

Macro library name                          Macros defined
------------------                          --------------
_$255$DUA28:[PLIRTL.OBJ]PLIRTMAC.MLB;1                   5
_$255$DUA28:[SYSLIB]STARLET.MLB;2                        6
TOTALS (all libraries)                                  11

513 GETS were required to define 11 macros.

There were no errors, warnings or information messages.

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

0308 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY



PLIFORMAT LIS

PLIGETBUF LIS

PLIMSGTXT LIS

PLIGETEDI LIS

PLIHEEP LIS

PLIPUTFIL LIS

PLIRMSBIS LIS

PLIRECOPT LIS

PLIOPEN LIS

PLIREAD LIS

PLIPROTEC LIS

PLIREWRIT LIS

PLIGETLIS LIS

PLIPUTEDI LIS

PLIPKDIVL LIS

PLIPUTLIS LIS

PLIMSGPTR LIS

PLIPKDIVS LIS

PLIPUTBUF LIS

PLIGETFIL LIS