```
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
```

```
PPPPPPPP    LL            IIIIII    RRRRRRRR    EEEEEEEEEE    AAAAAA    DDDDDDDD
PPPPPPPP    LL            IIIIII    RRRRRRRR    EEEEEEEEEE    AAAAAA    DDDDDDDD
PP      PP  LL              II      RR      RR  EE           AA      AA  DD      DD
PP      PP  LL              II      RR      RR  EE           AA      AA  DD      DD
PP      PP  LL              II      RR      RR  EE           AA      AA  DD      DD
PP      PP  LL              II      RR      RR  EE           AA      AA  DD      DD
PPPPPPPP    LL              II      RRRRRRRR    EEEEEEEE     AA      AA  DD      DD
PPPPPPPP    LL              II      RRRRRRRR    EEEEEEEE     AA      AA  DD      DD
PP          LL              II      RR  RR      EE           AAAAAAAAAA  DD      DD
PP          LL              II      RR  RR      EE           AAAAAAAAAA  DD      DD
PP          LL              II      RR    RR    EF           AA      AA  DD      DD      ....
PP          LL              II      RR    RR    EE           AA      AA  DD      DD      ....
PP          LLLLLLLLLL    IIIIII    RR      RR  EEEEEEEEEE   AA      AA  DDDDDDDD      ....
PP          LLLLLLLLLL    IIIIII    RR      RR  EEEEEEEEEE   AA      AA  DDDDDDDD      ....

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
```

```
0000     1          .title  pli$read - pl1 runtime read record
0000     2          .ident  /1-003/                                ; Edit WHM1003
0000     3
0000     4 ;
0000     5 ;*****************************************************************************
0000     6 ;*                                                                           *
0000     7 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     8 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     9 ;*  ALL RIGHTS RESERVED.                                                     *
0000    10 ;*                                                                           *
0000    11 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    12 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    13 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    14 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    15 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    16 ;*  TRANSFERRED.                                                             *
0000    17 ;*                                                                           *
0000    18 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    19 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    20 ;*  CORPORATION.                                                             *
0000    21 ;*                                                                           *
0000    22 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    23 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    24 ;*                                                                           *
0000    25 ;*                                                                           *
0000    26 ;*****************************************************************************
0000    27 ;
0000    28
0000    29 ;++
0000    30 ; facility:
0000    31 ;
0000    32 ;       VAX/VMS PL1 runtime library.
0000    33 ; abstract:
0000    34 ;
0000    35 ;       This module contains the pl1 runtime routines for reading a record
0000    36 ;       from a file.
0000    37 ;
0000    38 ; author: c. spitz 18-jul-79
0000    39 ;
0000    40 ; modified:
0000    41 ;
0000    42 ;       Bill Matthews 08-Jan-1981
0000    43 ;       V1.4-02:
0000    44 ;               Restore fcb_l_attr into r3 before calling pli$$keyto_r8.
0000    45 ;
0000    46 ;
0000    47 ;       1-003   Bill Matthews   29-September-1982
0000    48 ;
0000    49 ;               Invoke macros $defdat and rtshare instead of $defopr and share.
0000    50 ;
0000    51 ;--
0000    52
0000    53
0000    54 ;+
0000    55 ; external definitions
0000    56 ;-
0000    57
```

```
0000     58          $deffcb                         ;define file control block offsets
0000     59          $defdat                         ;define operand node data types
0000     60          $defpl1rtcons                   ;define pl1 runtime constants
0000     61          $fabdef                         ;define fab offsets
0000     62          $rabdef                         ;define rab offsets
0000     63          $rmsdef                         ;define rms error codes
0000     64
0000     65 ;+
0000     66 ; local definitions
0000     67 ;-
0000     68 $offset 4,positive,<-                    ;define arguments
0000     69          <fcbaddr,4>,-                   ;addr of fcb
0000     70          <readtype,4>,-                  ;read type
0000     71          <intoaddr,4>,-                  ;addr of into or set
0000     72          <intolen,2>,-                   ;length of into
0000     73          <intotyp,2>,-                   ;data type of into
0000     74          <keyaddr,4>,-                   ;addr of key or keyto
0000     75          <keylen,4>,-                    ;length of key
0000     76          <keytyp,4>,-                    ;data type of key
0000     77          <keynum,4>,-                    ;addr of key number
0000     78          <matchgtr,4>,-                  ;addr of match greater
0000     79          <matchgeq,4>,-                  ;addr of match greater equal
0000     80          <recidfrom,4>,-                 ;addr of record id from
0000     81          <recidto,4>,-                   ;addr of record id to
0000     82          <fxcaddr,4>,-                   ;addr of fixed control
0000     83          <fxclen,2>,-                    ;length of fixed control
0000     84          <fxctyp,2>,-                    ;data type of fixed control
0000     85          >                               ;
0000     86
0000     87          rtshare                         ;sharable
0000     88
```

```
                     0000     90
                     0000     91 ;++
                     0000     92 ; pli$read -- read a record from a file
                     0000     93 ;
                     0000     94 ; functional description:
                     0000     95 ;
                     0000     96 ; This routine reads a record from a pl1 file.
                     0000     97 ;
                     0000     98 ; inputs:
                     0000     99 ;       (ap) - number of argruments
                     0000    100 ;              7 if no options
                     0000    101 ;              14 if any options
                     0000    102 ;       4(ap) - addr of fcb
                     0000    103 ;       8(ap) - read type (bit 0 = 0 for into, 1 for set; bit 1 = 1 for key
                     0000    104 ;                          bit 2 = 1 for keyto
                     0000    105 ;              0 - into, no key, no keyto
                     0000    106 ;              1 - set, no key, no keyto
                     0000    107 ;              2 - into, key, no keyto
                     0000    108 ;              3 - set, key, no keyto
                     0000    109 ;              4 - into, no key, keyto
                     0000    110 ;              5 - set, no key, keyto
                     0000    111 ;       12(ap) - addr of into or set
                     0000    112 ;       16(ap) - length of into
                     0000    113 ;        18(ap) - data type of into
                     0000    114 ;       20(ap) - addr of key or keyto
                     0000    115 ;       24(ap) - size/prec of key or keyto
                     0000    116 ;       28(ap) - data type of key or keyto
                     0000    117 ;       32(ap) - addr of key number
                     0000    118 ;       36(ap) - addr of match greater
                     0000    119 ;       40(ap) - addr of match greater or equal
                     0000    120 ;       44(ap) - addr of record id from
                     0000    121 ;       48(ap) - addr of record id to
                     0000    122 ;       52(ap) - addr of fixed control
                     0000    123 ;       56(ap) - length of fixed control
                     0000    124 ;        58(ap) - data type of fixed control
                     0000    125 ;
                     0000    126 ; outputs:
                     0000    127 ;       fcb_l_attr
                     0000    128 ;              atr_m_delete, atr_m_currec, atr_m_virgin and atr_m_write are
                     0000    129 ;                      set to false
                     0000    130 ;       fcb_q_rfa is set to the rfa of the record read
                     0000    131 ;
                     0000    132 ; side effects:
                     0000    133 ;       if the file is closed, it is opened with the record, input and seql
                     0000    134 ;              attributes.
                     0000    135 ;       the record is read into the target.
                     0000    136 ;--
                     0000    137
                 01FC 0000   138         .entry  pli$read,^m<r2,r3,r4,r5,r6,r7,r8>
                     0002    139 ;
                     0002    140 ; check arguments. the read type must be >= 0 and <= 5. there must be
                     0002    141 ; either 7 or 14 arguments.
                     0002    142 ;
07   6C   D1 0002   143         cmpl    (ap),#7                 ;if not enough arguments
     05   18 0005   144         bgeq    10$                     ;then
     50   D4 0007   145         clrl    r0                      ;indicate not enough parms
  02B3    31 0009   146         brw     fail                    ;and fail
```

```
                52    04 AC    D0  000C   147 10$:    movl    fcbaddr(ap),r2          ;get address of fcb
                55    08 AC    D0  0010   148         movl    readtype(ap),r5         ;get read type
                         0A    18  0014   149         bgeq    30$                     ;if read type < 0
            50  00000000'8F    D0  0016   150 20$:    movl    #pli$_readop,r0         ;then set invalid read options
                      029F    31  001D   151         brw     fail                    ;and fail
                   05    55    D1  0020   152 30$:    cmpl    r5,#5                   ;if read option > 5
                         F1    14  0023   153         bgtr    20$                     ;then fail
                               0025   154 ;
                               0025   155 ; open the file if necessary. the file will be opened with the
                               0025   156 ; record attribute. if the file does not have the update attribute,
                               0025   157 ; from its declaration, input is also specified for the open. if
                               0025   158 ; the file is not opened after calling open, an error is signaled.
                               0025   159 ;
                53    0C A2    D0  0025   160         movl    fcb_l_attr(r2),r3       ;get files attributes
                2C 53    01    E0  0029   161         bbs     #atr_v_opened,r3,50$    ;if file not opened
              00001000 8F    DD  002D   162         pushl   #atr_m_record           ;then request record
                07 53    04    E0  0033   163         bbs     #atr_v_update,r3,40$    ;if update not specified
            6E  00000040 8F    C8  0037   164         bisl    #atr_m_input,(sp)       ;then also request input
                         52    DD  003E   165 40$:    pushl   r2                      ;push address of fcb
        00000000'GF      02    FB  0040   166         calls   #2,g^pli$open           ;open the file
                53    0C A2    D0  0047   167         movl    fcb_l_attr(r2),r3       ;get the new attributes
                0A 53    01    E0  004B   168         bbs     #atr_v_opened,r3,50$    ;if file still not opened
            50  00000000'8F    D0  004F   169         movl    #pli$_open,r0           ;then set open failure
                      0266    31  0056   170         brw     fail                    ;and fail
                               0059   171 ;
                               0059   172 ; make sure file has proper attributes. file must have record. if key
                               0059   173 ; or keyto specified, file must have keyed. file must not have output
                               0059   174 ; or delete specified.
                               0059   175 ;
                0A 53    0C    E0  0059   176 50$:    bbs     #atr_v_record,r3,60$    ;if file doesn't have record
            50  00000000'8F    D0  005D   177         movl    #pli$_notrec,r0         ;then set not record file
                      0258    31  0064   178         brw     fail                    ;and fail
                0A 53    05    E1  0067   179 60$:    bbc     #atr_v_output,r3,70$    ;if file has output
            50  00000000'8F    D0  006B   180         movl    #pli$_readout,r0        ;then set can't read output file
                      024A    31  0072   181         brw     fail                    ;and fail
                   02    55    D1  0075   182 70$:    cmpl    r5,#2                   ;if key or keyto specified
                         0E    19  0078   183         blss    80$                     ;then
                0A 53    08    E0  007A   184         bbs     #atr_v_keyed,r3,80$     ;if file not keyed
            50  00000000'8F    D0  007E   185         movl    #pli$_notkeyd,r0        ;then set not keyed file
                      0237    31  0085   186         brw     fail                    ;and fail
                               0088   187 ;
                               0088   188 ; 'deallocate' buffer, process options
                               0088   189 ;
        0C A2   00020000 8F    CA  0088   190 80$:    bicl    #atr_m_bfall,fcb_l_attr(r2) ;set buffer not allocated
                54    62 A2    9E  0090   191         movab   fcb_b_rab(r2),r4        ;get address of rab
        04 A4   00600000 8F    CA  0094   192         bicl    #<rab$m_kge!rab$m_kgt>,rab$l_rop(r4) ;clear match_gtr(_eql)
                   07    6C    D1  009C   193         cmpl    (ap),#7                 ;options passed?
                         46    13  009F   194         beql    100$                    ;if eql, then no
                   0E    6C    D1  00A1   195         cmpl    (ap),#14                ;enuf options passed?
                         0A    13  00A4   196         beql    90$                     ;if eql, then yes
            50  00000000'8F    D0  00A6   197         movl    #pli$_invnumopt,r0      ;set invalid options
                      020F    31  00AD   198         brw     fail                    ;and fail
                50    20 AC    D0  00B0   199 90$:    movl    keynum(ap),r0           ;get addr of key number option
              51    55   01    EF  00B4   200         extzv   #1,#1,r5,r1             ;set presence of key in r1
                00000000'GF    16  00B9   201         jsb     g^pli$$keynum           ;process key number
                50    24 AC    D0  00BF   202         movl    matchgtr(ap),r0         ;get addr of match greater option
                00000000'GF    16  00C3   203         jsb     g^pli$$matchgtr         ;process match greater
```

```
                50    28 AC    D0  00C9   204          movl    matchgeq(ap),r0          ;get addr of match greater or equal
          00000000'GF    16    0CD  00CD   205          jsb     g^pli$$matchgeq          ;process match greater or equal
                50    2C AC    D0  00D3   206          movl    recidfrom(ap),r0         ;get addr of record id from option
          00000000'GF    16    0D7  00D7   207          jsb     g^pli$$recidfrom         ;process record id from option
                50    30 AC    D0  00DD   208          movl    recidto(ap),r0           ;get addr of rfa to option
          00000000'GF    16    0E1  00E1   209          jsb     g^pli$$valrecidto        ;validate record id to
                                    00E7   210 ;
                                    00E7   211 ; process into option. copy the buffer size and address to the rab.
                                    00E7   212 ;
                   41 55    E8  00E7   213 100$:    blbs    r5,130$                  ;if into specified
             24 A4    0C AC    D0  00EA   214          movl    intoaddr(ap),rab$l_ubf(r4) ;then set read buffer address in rab
                   10 AC    DD  00EF   215          pushl   intolen(ap)              ;push size and data type
          00000000'GF    01    FB  00F2   216          calls   #1,g^pli$$bytesize       ;calculate byte size
                   03 50    E8  00F9   217          blbs    r0,110$                  ;if invalid data type
                    01C0    31  00FC   218          brw     fail                     ;then fail
                20 A4    51    B0  00FF   219 110$:    movw    r1,rab$w_usz(r4)         ;set byte size in rab
             12 AC    0057 8F    B1  0103   220          cmpw    #<dat_k_structure+64>,intotyp(ap) ;bit sized structure?
                         07    12  0109   221          bneq    115$                     ;if neq, no, cont
                   5E    51    C2  010B   222          subl    r1,sp                    ;get room for temp on stack
                24 A4    5E    D0  010E   223          movl    sp,rab$l_ubf(r4)         ;set addr of temp in rab
                12 AC    0B    B1  0112   224 115$:    cmpw    #dat_k_char_var,intotyp(ap) ;if data type = char var
                         45    12  0116   225          bneq    170$                     ;then
                   0C BC    B4  0118   226          clrw    @intoaddr(ap)            ;clear its length
                06 53    0D    E0  011B   227          bbs     #atr_v_scalvar,r3,120$   ;if scalar varying, read it all
                24 A4    02    C0  011F   228          addl    #2,rab$l_ubf(r4)         ;skip length field in address
                         38    11  0123   229          brb     170$                     ;continue
                20 A4    02    A0  0125   230 120$:    addw    #2,rab$w_usz(r4)         ;update size to include length of vcha
                         32    11  0129   231          brb     170$                     ;cont
                                    012B   232 ;
                                    012B   233 ; process pointer set. allocate a buffer if neccssary, to avoid excessive
                                    012B   234 ; overhead, we allocate it once per file opening. we 'deallocate' it by
                                    012B   235 ; clearing atr_m_bfall. we actually free the storage when the file is
                                    012B   236 ; closed.  the buffer size is the maximum record size from the fab, or the
                                    012B   237 ; default maximum record size.
                                    012B   238 ;
                   0C BC    D4  012B   239 130$:    clrl    @intoaddr(ap)            ;for pointer set, set pointer to 0
                   14 A2    D5  012E   240          tstl    fcb_l_buf(r2)            ;buffer already allocated?
                         20    12  0131   241          bneq    160$                     ;if neq, yes, cont
                   18 A2    DD  0133   242 140$:    pushl   fcb_l_buf_end(r2)        ;push size
                      5E    DD  0136   243          pushl   sp                       ;push address of temp
                   04 AE    DF  0138   244          pushal  4(sp)                    ;push address of size
          00000000'GF    02    FB  013B   245          calls   #2,g^lib$get_vm          ;allocate the buffer
                   0A 50    E8  0142   246          blbs    r0,150$                  ;if allocation failed
          50    00000000'8F    D0  0145   247          movl    #pli$_novirmem,r0        ;then set no virt. mem.
                    0170    31  014C   248          brw     fail                     ;and fail
                   14 A2 8ED0  014F   249 150$:    popl    fcb_l_buf(r2)            ;copy buffer address to fcb
             24 A4    14 A2    D0  0153   250 160$:    movl    fcb_l_buf(r2),rab$l_ubf(r4) ;copy buffer address to rab
             20 A4    18 A2    B0  0158   251          movw    fcb_l_buf_end(r2),rab$w_usz(r4) ;set size in rab
                                    015D   252 ;
                                    015D   253 ; process key option.
                                    015D   254 ;
                1C 55    01    E1  015D   255 170$:    bbc     #1,r5,190$               ;if key specified
                50    14 AC    D0  0161   256          movl    keyaddr(ap),r0           ;then set key buffer address in rab
                         0A    12  0165   257          bneq    180$                     ;if neq, cont
          50    00000000'8F    D0  0167   258          movl    #pli$_nokey,r0           ;set no key specified
                    014E    31  016E   259          brw     fail                     ;and fail
                50    14 AC    9E  0171   260 180$:    movab   keyaddr(ap),r0           ;point to key descr
```

```
                00000000'GF   16  0175   261          jsb     g^pli$$readkey_r6       ;process key
                         3B   11  0178   262          brb     240$                    ;continue
                              017D   263 ;
                              017D   264 ; sequential access is required if a key was not specified, or if the keyto
                              017D   265 ; option is present. make sure file has seql, and specify sequential access
                              017D   266 ; in the rab.
                              017D   267 ;
                      07   6C   D1  017D   268 190$:    cmpl    (ap),#7                 ;options passed?
                           05   15  0180   269          bleq    200$                    ;if leq, no
                        2C AC   D5  0182   270          tstl    recidfrom(ap)           ;record id from specified?
                           31   12  0185   271          bneq    240$                    ;if neq, yes, continue
                   0A 53   0A   E0  0187   272 200$:    bbs     #atr_v_seql,r3,210$     ;if file doesn't have seql
               50  00000000'8F   D0  018B   273          movl    #pli$_notsql,r0         ;then set not seql file
                        012A   31  0192   274          brw     fail                    ;and fail
                05 00BC C2   05   E1  0195   275 210$:    bbc     #fab$v_bio,<fcb_b_fab+fab$b_fac>(r2),220$ ;if block io
                        38 A2   D4  019B   276          clrl    rab$l_bkt(r2)           ;set for seql access
                           18   11  019E   277          brb     240$                    ;cont
                   10 53   14   E1  01A0   278 220$:    bbc     #atr_v_write,r3,230$    ;if last oper was a write
                10 A4   20 A2   7D  01A4   279          movq    fcb_q_rfa(r2),rab$w_rfa(r4) ;set correct rfa in rab
                           09   13  01A9   280          beql    230$                    ;if eql, its a term so cont
                   1E A4   02   90  01AB   281          movb    #rab$c_rfa,rab$b_rac(r4) ;set for rfa access in rab
                           53   D4  01AF   282          clrl    r3                      ;set no key specified
                        0156   30  01B1   283          bsbw    pli$$smallget           ;point to the record written
                   1E A4   00   90  01B4   284 230$:    movb    #rab$c_seq,rab$b_rac(r4) ;set for seq access in rab
                              01B8   285 ;
                              01B8   286 ; get the record
                              01B8   287 ;
                      2C A4   D4  01B8   288 240$:    clrl    rab$l_rhb(r4)           ;assume no fixed control to
                   07   6C   D1  01BB   289          cmpl    (ap),#7                 ;options passed?
                           0A   13  01BE   290          beql    250$                    ;if eql, no
                50   34 AC   9E  01C0   291          movab   fxcaddr(ap),r0          ;get addr of fixed control descr
                00000000'GF   16  01C4   292          jsb     g^pli$$fxctlto_r6       ;process fixed control
                53   14 AC   9E  01CA   293 250$:    movab   keyaddr(ap),r3          ;set addr of key
             16 00BC C2   05   E1  01CE   294          bbc     #fab$v_bio,<fcb_b_fab+fab$b_fac>(r2),270$ ;if block io
                              01D4   295          $read   r4                      ;do a read
                   15 50   E8  01DD   296 260$:    blbs    r0,290$                 ;if lbs, cont
                50   00000000'8F   D0  01E0   297          movl    #pli$_rmsr,r0           ;set error code in rab
                        00D5   31  01E7   298          brw     fail                    ;and fail
                              01EA   299 270$:    $get    r4                      ;get the record
                           E8   11  01F3   300 280$:    brb     260$                    ;cont
                              01F5   301 ;
                              01F5   302 ; if into is bit sized structure, copy from temp to target
                              01F5   303 ;
                   1C A2   22 A4   3C  01F5   304 290$:    movzwl  rab$w_rsz(r4),fcb_l_buf_pt(r2) ;save size of record read
                      50 08 AC   E8  01FA   305          blbs    readtype(ap),310$       ;if read set, cont
                12 AC   0057 8F   B1  01FE   306          cmpw    #<dat_k_structure+64>,intotyp(ap) ;bit sized structure?
                           1A   12  0204   307          bneq    295$                    ;if neq, no, cont
                           3C   BB  0206   308          pushr   #^m<r2,r3,r4,r5>        ;save regs
                50   20 A4   01   A3  0208   309          subw3   #1,rab$w_usz(r4),r0     ;get number of whole bytes
             0C BC   24 B4   50   28  020D   310          movc3   r0,@rab$l_ubf(r4),@intoaddr(ap) ;copy whole bytes
                  10 AC   F8 8F   8A  0213   311          bicb    #^c7,intolen(ap)        ;get number of bits left
           63   10 AC   00   61   F0  0218   312          insv    (r1),#0,intolen(ap),(r3) ;copy remaining bits
                           3C   BA  021E   313          popr    #^m<r2,r3,r4,r5>        ;restore regs
                              0220   314 ;
                              0220   315 ; set size of record read if char var
                              0220   316 ;
                      12 AC   0B   B1  0220   317 295$:    cmpw    #dat_k_char_var,intotyp(ap) ;reading char var?
```

```
                     0C   12  0224   318          bneq    300$                            ;if neq, no, cont
          07 0C A2   0D   E0  0226   319          bbs     #atr_v_scalvar,fcb_l_attr(r2),300$ ;if scalvar, length field was
                             022B   320                                                  ;read, cont
          0C BC   22 A4   B0  022B   321          movw    rab$w_rsz(r4),@intoaddr(ap) ;plug size read in length field
                     1C   11  0230   322          brb     310$                            ;cont
          20 A4   22 A4   B1  0232   323 300$:    cmpw    rab$w_rsz(r4),rab$w_usz(r4) ;was record and target same size?
                     15   13  0237   324          beql    310$                            ;if eql, then yes, cont
                     52   DD  0239   325          pushl   r2                              ;set fcb addr
            00000000'8F   DD  023B   326          pushl   #pli$_record                    ;set record error
            00000000'8F   DD  0241   327          pushl   #<pli$_error&^c7>               ;set error condition(warning)
        00000000'GF   03   FB  0247   328          calls   #3,g^pli$io_error               ;signal the condition and continue
                             024E   329 ;
                             024E   330 ; if fixed control to is a bit sized structure, copy from temp to target
                             024E   331 ;
                07   6C   D1  024E   332 310$:    cmpl    (ap),#7                         ;options passed?
                     2B   13  0251   333          beql    319$                            ;if eql, no, cont
                  34 AC   D5  0253   334          tstl    fxcaddr(ap)                     ;fixed control specified?
                     26   13  0256   335          beql    319$                            ;if eql, no, cont
          3A AC   0057 8F   B1  0258   336          cmpw    #<dat_k_structure+64>,fxctyp(ap) ;is it a bit sized structure?
                     1E   12  025E   337          bneq    319$                            ;if neq, no, cont
             50   00E5 C2   9A  0260   338          movzbl  <fcb_b_fab+fab$b_fsz>(r2),r0 ;get fixed control size
                     17   13  0265   339          beql    319$                            ;if eql, none, cont
                     3C   BB  0267   340          pushr   #^m<r2,r3,r4,r5>                ;save regs
                     50   D7  0269   341          decl    r0                              ;get number of whole bytes to copy
       34 BC    2C B4   50   28  026B   342          movc3   r0,@rab$l_rhb(r4),@fxcaddr(ap) ;copy whole bytes
             38 AC    F8 8F   8A  0271   343          bicb    #^c7,fxclen(ap)             ;get number of bits remaining
    63    38 AC   00   61   F0  0276   344          insv    (r1),#0,fxclen(ap),(r3) ;copy remaining bits
                     3C   BA  027C   345          popr    #^m<r2,r3,r4,r5>                ;restore regs
                             027E   346 ;
                             027E   347 ; process keyto option.
                             027E   348 ;
          0E 08 AC   02   E1  027E   349 319$:    bbc     #2,readtype(ap),320$    ;if keyto specified
             50   14 AC   9E  0283   350          movab   keyaddr(ap),r0          ;set addr of keyto descr
             53   0C A2   D0  0287   351          movl    fcb_l_attr(r2),r3       ;load the attributes into r3 for routine cal
            00000000'GF   16  028B   352          jsb     g^pli$$keyto_r8         ;process keyto
                             0291   353 ;
                             0291   354 ; process pointer set
                             0291   355 ;
             0D 08 AC   E9  0291   356 320$:    blbc    readtype(ap),330$       ;if pointer set specified
    0C A2    00020000 8F   C8  0295   357          bisl    #atr_m_bfall,fcb_l_attr(r2) ;then mark buffer allocated
          0C BC   14 A2   D0  029D   358          movl    fcb_l_buf(r2),@intoaddr(ap) ;and plug pointer set
                             02A2   359 ;
                             02A2   360 ; successful completion
                             02A2   361 ;
                07   6C   D1  02A2   362 330$:    cmpl    (ap),#7                 ;options passed?
                     0A   15  02A5   363          bleq    340$                    ;if leq, no
             50   30 AC   D0  02A7   364          movl    recidto(ap),r0          ;get addr of record id to option
                     04   13  02AB   365          beql    340$                    ;if eql, not specified
             60   10 A4   7D  02AD   366          movq    rab$w_rfa(r4),(r0)      ;set record id to
    0C A2    021C0000 8F   CA  02B1   367 340$:    bicl    #<atr_m_delete!atr_m_currec! - ;clear delete and indicate
                             02B9   368                  atr_m_virgin!atr_m_write>,fcb_l_attr(r2) ;current record
                             02B9   369                                          ;correct, not virgin, not write
          20 A2   10 A4   7D  02B9   370          movq    rab$w_rfa(r4),fcb_q_rfa(r2) ;set correct rfa in fcb
                     04   02BE   371          ret                             ;return
                             02BF   372
```

```
                                02BF   374
                     12   12    02BF   375 fail:   bneq    10$                         ;if not enough parms
          50  00000000'8F   D0   02C1   376         movl    #pli$_parm,r0               ;then specify not enough parameters
                     52   D4   02C8   377         clrl    r2                          ;assume no fcb specified
                        01   6C   D1   02CA   378         cmpl    (ap),#1                     ;if fcb specified
                     29   19    02CD   379         blss    40$                         ;then
                  52   04  AC   D0   02CF   380         movl    fcbaddr(ap),r2              ;get address of fcb
          50  00000000'8F   D1   02D3   381 10$:     cmpl    #pli$_rmsr,r0               ;if error code in rms rab
                     1C   12    02DA   382         bneq    40$                         ;and
    08 A4  0001827A 8F   D1   02DC   383         cmpl    #rms$_eof,rab$l_sts(r4)     ;if end of file
                     0C   12    02E4   384         bneq    20$                         ;then
                     52   DD   02E6   385         pushl   r2                          ;set fcb addr
                     50   DD   02E8   386         pushl   r0                          ;set error code
                00000000'8F   DD   02EA   387         pushl   #pli$_endfile               ;set endfile condition
                     10   11   02F0   388         brb     50$                         ;else
                00000000'GF   16   02F2   389 20$:     jsb     g^pli$$chk_keycnd           ;check for key condition
                     52   DD   02F8   390 40$:     pushl   r2                          ;set fcb addr
                     50   DD   02FA   391         pushl   r0                          ;set error code
                00000000'8F   DD   02FC   392         pushl   #pli$_error                 ;set error condition
          00000000'GF   03   FB   0302   393 50$:     calls   #3,g^pli$io_error           ;signal the condition
                        04   0309   394         ret                                 ;and return
                                030A   395
                                030A   396 ;++
                                030A   397 ;pli$$smallget
                                030A   398 ; this routine saves the current user buffer address in the rab, allocates
                                030A   399 ; a 4 byte buffer on the stack and issues a $get. if the get fails because
                                030A   400 ; of buffer too small, or if it succeeds, the routine returns successfully.
                                030A   401 ; otherwise, the pl/i error condition is signaled, with the appropriate
                                030A   402 ; subcode. This routine is used to correctly position rms for keyed access.
                                030A   403 ; A $get is required because the $find operation does not affect rms's next
                                030A   404 ; record.
                                030A   405 ;
                                030A   406 ; inputs:
                                030A   407 ;       r3 - address of key descr for onkey
                                030A   408 ;       r4 - address of rab
                                030A   409 ; outputs:
                                030A   410 ;       none
                                030A   411 ; side effects:
                                030A   412 ;       the file is positioned to the requested rfa
                                030A   413 ;--
                                030A   414
                                030A   415 pli$$smallget::
                                030A   416         assume  <rab$w_usz+2> eq rab$w_rsz
                  20  A4   DD   030A   417         pushl   rab$w_usz(r4)               ;save buffer sizes
                  24  A4   DD   030D   418         pushl   rab$l_ubf(r4)               ;save user buffer addr
                  28  A4   DD   0310   419         pushl   rab$l_rbf(r4)               ;save record buffer addr
               24 A4    7E   DE   0313   420         moval   -(sp),rab$l_ubf(r4)         ;set legal buffer addr
               20 A4    04   B0   0317   421         movw    #4,rab$w_usz(r4)            ;set length of zero
                                031B   422         $get    r4                          ;get the record
                  13  50   E8   0324   423         blbs    r0,10$                      ;if lbs, cont
    000181A8 8F    50   D1   0327   424         cmpl    r0,#rms$_rtb                ;record too big?
                     0A   13   032E   425         beql    10$                         ;if eql, yes, treat as success
    50  00000000'8F   D0   0330   426         movl    #pli$_rmsr,r0               ;set rms rab error
                  FF85   31   0337   427         brw     fail                        ;and fail
               5E  04  AE   9E   033A   428 10$:     movab   4(sp),sp                    ;clean stack
                  28  A4 8ED0   033E   429         popl    rab$l_rbf(r4)               ;restore record buffer addr
                  24  A4 8ED0   0342   430         popl    rab$l_ubf(r4)               ;restore user buffer addr
```

```
20 A4 8ED0  0346   431          popl    rab$w_usz(r4)           ;restore buffer sizes
        05  034A   432          rsb                             ;return
            034B   433
            034B   434          .end
```

```
$$.TMP1                 = 00000001
$$.TMP2                 = 00000054
ATR_M_BFALL             = 00020000
ATR_M_CURREC            = 00040000
ATR_M_DELETE            = 00080000
ATR_M_INPUT             = 00000040
ATR_M_RECORD            = 00001000
ATR_M_VIRGIN            = 02000000
ATR_M_WRITE             = 00100000
ATR_V_KEYED             = 00000008
ATR_V_OPENED            = 00000001
ATR_V_OUTPUT            = 00000005
ATR_V_RECORD            = 0000000C
ATR_V_SCALVAR           = 0000000D
ATR_V_SEQL              = 0000000A
ATR_V_UPDATE            = 00000004
ATR_V_WRITE             = 00000014
DAT_K_CHAR_VAR          = 0000000B
DAT_K_STRUCTURE         = 00000017
DIR...                  = 00000001
FAB$B_FAC               = 00000016
FAB$B_FSZ               = 0000003F
FAB$V_BIO               = 00000005
FAIL                      000002BF R      02
FCBADDR                   00000004
FCB_B_ENVIR               000001C2
FCB_B_ESA                 0000012E
FCB_B_EXTRA               0000003D
FCB_B_FAB                 000000A6
FCB_B_IDENT               00000040
FCB_B_IDENT_NAM           00000042
FCB_B_NAM                 000000F6
FCB_B_NUMKCBS             0000003C
FCB_B_RAB                 00000062
FCB_C_LEN                 000001C2
FCB_C_STRLEN              00000034
FCB_L_ATTR                0000000C
FCB_L_BUF                 00000014
FCB_L_BUF_END             00000018
FCB_L_BUF_PT              0000001C
FCB_L_CNDADDR             000001B2
FCB_L_CONDIT              000001AE
FCB_L_DTTR                00000010
FCB_L_ERROR               00000008
FCB_L_KCB                 00000038
FCB_L_NEXT                00000000
FCB_L_PREVIOUS            00000004
FCB_L_PRN                 00000034
FCB_Q_RFA                 00000020
FCB_W_COLUMN              0000002E
FCB_W_IDENT_LEN           00000040
FCB_W_LINE                00000030
FCB_W_LINESIZE            0000002A
FCB_W_PAGE                00000032
FCB_W_PAGESIZE            0000002C
FCB_W_REVISION            00000028
FXCADDR                   00000034
FXCLEN                    00000038
FXCTYP                    0000003A
INTOADDR                  0000000C
INTOLEN                   00000010
INTOTYP                   00000012
KEYADDR                   00000014
KEYLEN                    00000018
KEYNUM                    00000020
KEYTYP                    0C00001C
LIB$GET_VM                ********    X   02
MATCHGEQ                  00000028
MATCHGTR                  00000024
PLI$$BYTESIZE             ********    X   02
PLI$$CHK_KEYCND           ********    X   02
PLI$$FXCTLTO_R6           ********    X   02
PLI$$KEYNUM               ********    X   02
PLI$$KEYTO_R8             ********    X   02
PLI$$MATCHGEQ             ********    X   02
PLI$$MATCHGTR             ********    X   02
PLI$$READKEY_R6           ********    X   02
PLI$$RECIDFROM            ********    X   02
PLI$$SMALLGET             0000030A RG     02
PLI$$VALRECIDTO           ********    X   02
PLI$IO_ERROR              ********    X   02
PLI$OPEN                  ********    X   02
PLI$READ                  00000000 RG     02
PLI$_ENDFILE              ********    X   02
PLI$_ERROR                ********    X   02
PLI$_INVNUMOPT            ********    X   02
PLI$_NOKEY                ********    X   02
PLI$_NOTKEYD              ********    X   02
PLI$_NOTREC               ********    X   02
PLI$_NOTSQL               ********    X   02
PLI$_NOVIRMEM             ********    X   02
PLI$_OPEN                 ********    X   02
PLI$_PARM                 ********    X   02
PLI$_READOP               ********    X   02
PLI$_READOUT              ********    X   02
PLI$_RECORD               ********    X   02
PLI$_RMSR                 ********    X   02
RAB$B_RAC               = 0000001E
RAB$C_RFA               = 00000002
RAB$C_SEQ               = 00000000
RAB$L_BKT               = 00000038
RAB$L_RBF               = 00000028
RAB$L_RHB               = 0000002C
RAB$L_ROP               = 00000004
RAB$L_STS               = 00000008
RAB$L_UBF               = 00000024
RAB$M_KGE               = 00200000
RAB$M_KGT               = 00400000
RAB$W_RFA               = 00000010
RAB$W_RSZ               = 00000022
RAB$W_USZ               = 00000020
READTYPE                  00000008
RECIDFROM                 0000002C
RECIDTO                   00000030
```

```
RMS$_EOF                        = 0001827A
RMS$_RTB                        = 000181A8
SIZ...                          = 00000001
SYS$GET                           ********  GX   02
SYS$READ                          ********  GX   02
```

Psect synopsis

PSECT name	Allocation	PSECT No.	Attributes
. ABS .	00000000 (0.)	00 (0.)	NOPIC USR CON ABS LCL NOSHR NOEXE NORD NOWRT NOVEC BYTE
ABS	000001C2 (450.)	01 (1.)	NOPIC USR CON ABS LCL NOSHR EXE RD WRT NOVEC BYTE
_PLI$CODE	0000034B (843.)	02 (2.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	9	00:00:00.07	00:00:00.28
Command processing	94	00:00:00.55	00:00:02.24
Pass 1	212	00:00:08.01	00:00:18.96
Symbol table sort	5	00:00:00.89	00:00:01.81
Pass 2	85	00:00:01.81	00:00:03.79
Symbol table output	14	00:00:00.12	00:00:00.22
Psect synopsis output	2	00:00:00.03	00:00:00.03
Cross-reference output	0	00:00:00.00	00:00:00.00
Assembler run totals	421	00:00:11.48	00:00:27.33

The working set limit was 1050 pages.
44570 bytes (88 pages) of virtual memory were used to buffer the intermediate code.
There were 40 pages of symbol table space allocated to hold 715 non-local and 42 local symbols.
434 source lines were read in Pass 1, producing 16 object records in Pass 2.
23 pages of virtual memory were used to define 20 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[PLIRTL.OBJ]PLIRTMAC.MLB;1	6
_255DUA28:[SYSLIB]STARLET.MLB;2	11
TOTALS (all libraries)	17

808 GETS were required to define 17 macros.

There were no errors, warnings or information messages.

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

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