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

FILEID**PLIRECOPT

```
PPPPPPPP    LL          IIIIII   RRRRRRRR   EEEEEEEEEE   CCCCCCCC   OOOOOO   PPPPPPPP   TTTTTTTTTT
PPPPPPPP    LL          IIIIII   RRRRRRRR   EEEEEEEEEE   CCCCCCCC   OOOOOO   PPPPPPPP   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
PPPPPPPP    LL            II     RRRRRRRR   EEEEEEEE   CC         OO      OO PPPPPPPP       TT
PPPPPPPP    LL            II     RRRRRRRR   EEEEEEEE   CC         OO      OO PPPPPPPP       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
```

```
0000     1          .title  pli$recopt - pl1 runtime record io option processing
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 ;++
0000    31 ; facility:
0000    32 ;
0000    33 ;       VAX/VMS PL1 runtime library.
0000    34 ; abstract:
0000    35 ;
0000    36 ;       This module contains the pl1 runtime routines for processing the
0000    37 ;       common options for record i/o statements.
0000    38 ;
0000    39 ; author: c. spitz 7-feb-80
0000    40 ;
0000    41 ; modified:
0000    42 ;
0000    43 ;       Bill Matthews 22-Sep-81
0000    44 ;
0000    45 ;               Fix so that routine pli$$recidfrom does return with the
0000    46 ;               Z condition code bit clear if an rfa is specified.
0000    47 ;
0000    48 ;
0000    49 ;       1-003   Bill Matthews   29-September-1982
0000    50 ;
0000    51 ;               Invoke macros $defdat and rtshare instead of $defopr and share.
0000    52 ;
0000    53 ;--
0000    54
0000    55
0000    56 ;
0000    57 ; external definitions
```

```
0000     58 ;
0000     59          $deffcb                         ;define file control block offsets
0000     60          $defstk                         ;define runtime stack
0000     61          $defdat                         ;define operand data types
0000     62          $defkcb                         ;define key control block offsets
0000     63          $fabdef                         ;define fab offsets
0000     64          $rabdef                         ;define rab offsets
0000     65
0000     66 ;
0000     67 ; local data
0000     68 ;
0000     69
0000     70          rtshare                         ;sharable
0000     71
0000     72
0000     73
```

```
                     0000     75 ;++
                     0000     76 ;pli$$keyto
                     0000     77 ; this routine uses the current key number to determine the data type of
                     0000     78 ; the key of reference and its position in the record. it allocates space
                     0000     79 ; on the stack for a temporary, if the key is segmented, and 'unsegments'
                     0000     80 ; the key. finally, the key is converted to the keyto destination.
                     0000     81 ;       r0 - addr of keyto descr
                     0000     82 ;               0(r0) - addr of keyto
                     0000     83 ;               4(r0) - size/prec of keyto
                     0000     84 ;               8(r0) - data type of keyto
                     0000     85 ;       r2 - addr of fcb
                     0000     86 ;       r3 - fcb_l_attr
                     0000     87 ;       r4 - addr of rab
                     0000     88 ; outputs:
                     0000     89 ;       r5,r6,r7,r8 - destroyed
                     0000     90 ;--
                     0000     91
                     0000     92         .enabl  lsb
                     0000     93 pli$$keyto_r8::
          56    6E   D0 0000     94         movl    (sp),r6                 ;save return addr
77 00BC   C2    05   E0 0003     95         bbs     #fab$v_bio,<fcb_b_fab+fab$b_fac>(r2),140$ ;if blockio, cont
      68 53    10   E1 0009     96         bbc     #atr_v_indexed,r3,110$  ;if indexed,
      55    35 A4   9A 000D     97         movzbl  rab$b_krf(r4),r5        ;get current key of ref
      55    2C C4   11 0011     98         mull    #kcb_c_len,r5           ;get index to correct kcb
      55    38 A2   C0 0014     99         addl    fcb_l_kcb(r2),r5        ;
         65    0A D1   0018    100         cmpl    #dat_k_char,kcb_l_dtyp(r5) ;character key?
               2B 12   001B    101         bneq    60$                     ;if neq, no
5E   000000FC 8F C2   001D    102         subl    #252,sp                 ;get room for largest key possible
               5E DD   0024    103         pushl   sp                      ;set addr of temp
               3D BB   0026    104         pushr   #^m<r0,r2,r3,r4,r5>     ;save regs
      53    18 AE   9E 0028    105         movab   24(sp),r3               ;get start addr of temp
      57    24 A4   D0 002C    106         movl    rab$l_ubf(r4),r7        ;get addr of buffer
      58    08 A5   9E 0030    107         movab   kcb_w_pos0(r5),r8       ;get addr of pos0 in kcb
         51    88 3C   0034    108 10$:    movzwl  (r8)+,r1                ;get position in record
         51    57 C0   0037    109         addl    r7,r1                   ;add in addr of buffer
               68 B5   003A    110         tstw    (r8)                    ;any thing left?
               06 13   003C    111         beql    20$                     ;if eql, no
   63    61    88 28   003E    112         movc3   (r8)+,(r1),(r3)         ;copy this segment
               F0 11   0042    113         brb     10$                     ;go again
               3D BA   0044    114 20$:    popr    #^m<r0,r2,r3,r4,r5>     ;restore regs
               08 11   0046    115         brb     100$                    ;cont
   7E    08 A5   3C 0048    116 60$:    movzwl  kcb_w_pos0(r5),-(sp)    ;push position in buffer of key
   6E    24 A4   C0 004C    117         addl    rab$l_ubf(r4),(sp)      ;add in buffer addr
               00 DD   0050    118 100$:   pushl   #0                      ;set dst offset
            04 A0 DD   0052    119         pushl   4(r0)                   ;set dst prec
            08 A0 DD   0055    120         pushl   8(r0)                   ;set dst data type
               50 DD   0058    121         pushl   r0                      ;set addr of addr of dst
               00 DD   005A    122         pushl   #0                      ;set src offset
18 00BC   C2    05   E0 005C    123         bbs     #fab$v_bio,<fcb_b_fab+fab$b_fac>(r2),130$ ;if blockio, cont
   14 53    10   E1 0062    124         bbc     #atr_v_indexed,r3,130$  ;if indexed
            04 A5 DD   0066    125         pushl   kcb_l_prec(r5)          ;set src prec
            65 DD   0069    126         pushl   kcb_l_dtyp(r5)          ;set src data type
            1C AE   9F 006B    127 120$:   pushab  28(sp)                  ;set addr of addr of src
               08 DD   006E    128         pushl   #8                      ;set number of args
             00CD 30   0070    129         bsbw    key_cvrt                ;convert key into keyto, checking error
               66 17   0073    130 sigerr: jmp     (r6)                    ;return
            38 A4   9F 0075    131 110$:   pushab  rab$l_bkt(r4)           ;set addr of key
```

```
                     D6   11  0078   132                brb     100$                    ;cont
                     1F   DD  007A   133 130$:           pushl   #31                     ;set prec of src
                     02   DD  007C   134                 pushl   #dat_k_fix_bin          ;set data type of src
                     EB   11  007E   135                 brb     120$                    ;cont
                  10 A4   9F  0080   136 140$:           pushab  rab$w_rfa(r4)           ;set addr of key
                     CB   11  0083   137                 brb     100$                    ;cont
                              0085   138                 .dsabl  lsb
                              0085   139
                              0085   140 ;++
                              0085   141 ; pli$$key_hnd
                              0085   142 ; this routine is the condition handler for pli$$readkey and pli$writekey.
                              0085   143 ; it is jsb'd to from an entry mask that lies within the file control
                              0085   144 ; block. This vectoring is necessary, in order to easily get the address of the
                              0085   145 ; file control block.  After calculating the address of the file control
                              0085   146 ; block, this routine sets the error code in fab$l_error.
                              0085   147 ;
                              0085   148 ; inputs:
                              0085   149 ;       0(sp) - address of byte following jsb instruction in fcb.
                              0085   150 ; outputs:
                              0085   151 ;       none
                              0085   152 ;--
                              0085   153
                              0085   154 pli$$key_hnd::
     50   6E   000001B6 8F    C3  0085   155                 subl3   #fcb_l_cndaddr+4,(sp),r0 ;get fcb address
               6E   01AE C0    9E  008D   156                 movab   fcb_l_condit(r0),(sp)   ;get addr of handler
                    50   5D    D0  0092   157                 movl    fp,r0                   ;start with this frame
                    52   50    D0  0095   158 10$:            movl    r0,r2                   ;save stack frame
                  50    0C A0    D0  0098   159                 movl    stk_l_fp(r0),r0         ;address next frame
                 60    10   03    0D  009C   160                 probew  #3,#stk_l_pc,(r0)       ;frame accessible?
                           18   13  00A0   161                 beql    30$                     ;if eql then end of stack found, return
                      60   6E    D1  00A2   162                 cmpl    (sp),stk_l_cnd_hnd(r0)  ;this frame's handler us?
                           EE   12  00A5   163                 bneq    10$                     ;if neq, no, look at next frame
                           60   D4  00A7   164                 clrl    stk_l_cnd_hnd(r0)       ;we aren't needed anymore
                      51   50    D0  00A9   165                 movl    r0,r1                   ;set addr of frame to return to
                   50    10 A2    D0  00AC   166                 movl    stk_l_pc(r2),r0         ;set addr of pc to return to
                  08 A2    5E    D0  00B0   167                 movl    sp,stk_l_ap(r2)         ;set non-zero for status
              00000000'GF   17  00B4   168                 jmp     g^pli$goto              ;return
     50   00000000'8F    D0  00BA   169 30$:            movl    #ss$_resignal,r0        ;set to resignal
                           04  00C1   170                 ret                             ;return
                              00C2   171
                              00C2   172 ;++
                              00C2   173 ;pli$$readkey
                              00C2   174 ; this routine uses the current key number to determine the data type of
                              00C2   175 ; the key of reference. it then allocates space on the stack to store a
                              00C2   176 ; temporary of this data type. it converts the source key to the temporary
                              00C2   177 ; and stores the key address and size in the rab.
                              00C2   178 ; inputs:
                              00C2   179 ;       r0 - addr of key descr
                              00C2   180 ;               0(r0) - addr of key
                              00C2   181 ;               4(r0) - size/prec of key
                              00C2   182 ;               8(r0) - data type of key
                              00C2   183 ;       r2 - addr of fcb
                              00C2   184 ;       r3 - fcb_l_attr
                              00C2   185 ;       r4 - addr of rab
                              00C2   186 ; outputs:
                              00C2   187 ;       r6 is destroyed
                              00C2   188 ;--
```

```
                                00C2   189
                                00C2   190 pli$$readkey_r6::
                          56    6E   D0  00C2   191          movl    (sp),r6                         ;save return addr
                       3C 53    10   E1  00C5   192          bbc     #atr_v_indexed,r3,100$          ;if indexed,
                       51   35  A4   9A  00C9   193          movzbl  rab$b_krf(r4),r1                ;get current key of ref
                       3C A2    51   91  00CD   194          cmpb    r1,fcb_b_numkcbs(r2)            ;legal key number?
                                0A   1B  00D1   195          blequ   10$                             ;if lequ, yes
                    50  00000000'8F   D0  00D3   196          movl    #pli$_invindnum,r0              ;set invalid key number
                              0378   31  00DA   197          brw     fail                            ;and fail
                          51    2C   C4  00DD   198 10$:     mull    #kcb_c_len,r1                   ;get index to correct kcb
                       51   38  A2   C0  00E0   199          addl    fcb_l_kcb(r2),r1                ;
                    5E  000000FC 8F   C2  00E4   200          subl    #252,sp                         ;get room for largest key possible
                                5E   DD  00EB   201          pushl   sp                              ;set addr of dst
                                00   DD  00ED   202          pushl   #0                              ;set dst offset to zero
                                   04 A1   DD  00EF   203          pushl   kcb_l_prec(r1)                  ;set dst prec
                                61   DD  00F2   204          pushl   kcb_l_dtyp(r1)                  ;set dst data type
                          34 A4    0A   A1  90  00F4   205          movb    kcb_w_len0(r1),rab$b_ksz(r4) ;set size of key in rab if not char
                                61  0A   D1  00F9   206          cmpl    #dat_k_char,kcb_l_dtyp(r1) ;character key?
                                13   12  00FC   207          bneq    110$                            ;if neq, no
                          34 A4    04   A1  90  00FE   208          movb    kcb_l_prec(r1),rab$b_ksz(r4) ;set size of key in rab
                                0C   11  0103   209          brb     110$                            ;cont in common
                                5E   DD  0105   210 100$:    pushl   sp                              ;set addr of dst
                                00   DD  0107   211          pushl   #0                              ;set dst offset to zero
                                1F   DD  0109   212          pushl   #31                             ;set dst prec
                                02   DD  010B   213          pushl   #dat_k_fix_bin                  ;set dst data type
                             34 A4   04   90  010D   214          movb    #4,rab$b_ksz(r4)                ;set size of key in rab
                                   0C AE   9F  0111   215 110$:    pushab  12(sp)                          ;set dst addr
                                00   DD  0114   216          pushl   #0                              ;set src offset
                                   04 A0   DD  0116   217          pushl   4(r0)                           ;set src prec
                                   08 A0   DD  0119   218          pushl   8(r0)                           ;set src data type
                                50   DD  011C   219          pushl   r0                              ;set addr of addr of src
                             53    50   D0  011E   220          movl    r0,r3                           ;copy key descr addr of onkey
                                08   DD  0121   221          pushl   #8                              ;set number of args
                                  001A   30  0123   222          bsbw    key_cvrt                        ;convert src to key data type
                          5E    28 AE   9E  0126   223          movab   40(sp),sp                       ;clean stack
                    0A 00BC  C2    05   E0  012A   224          bbs     #fab$v_bio,<fcb_b_fab+fab$b_fac>(r2),130$ ;if blockio, cont
                          30 A4    5E   D0  0130   225          movl    sp,rab$l_kbf(r4)                ;set key buffer addr in rab
                          1E A4    01   90  0134   226          movb    #rab$c_key,rab$b_rac(r4)        ;set keyed access in rab
                                66   17  0138   227          jmp     (r6)                            ;return
                                   38 A4 8ED0  013A   228 130$:    popl    rab$l_bkt(r4)                   ;set virt block num for block io
                                66   17  013E   229          jmp     (r6)                            ;return;
                                     0140   230
                                     0140   231 key_cvrt:
                                5C   DD  0140   232          pushl   ap                              ;save ap
                                5C   D4  0142   233          clrl    ap                              ;set up status temp for handler
                             6D    01AE C2  9E  0144   234          movab   fcb_l_condit(r2),(fp)           ;set up condition handler
                    00000000'GF    08 AE   FA  0149   235          callg   8(sp),g^pli$cvrt_any            ;do the conversion
                                6D   D4  0151   236          clrl    (fp)                            ;we don't need handler anymore
                                5C   D5  0153   237          tstl    ap                              ;conversion fail?
                                04   12  0155   238          bneq    10$                             ;if neq, yes, do onkey
                                5C 8ED0  0157   239          popl    ap                              ;restore ap
                                     05  015A   240          rsb                                     ;return
                          50    FF14 CF   9E  015B   241 10$:     movab   sigerr,r0                       ;get return addr for keyto conversion
                             50    04 AE   D1  0160   242          cmpl    4(sp),r0                        ;should we signal error?
                                0A   12  0164   243          bneq    20$                             ;if neq, no, do onkey
                    50  00000000'8F   D0  0166   244 15$:     movl    #pli$_cnverr,r0                 ;set conversion error
                              02E5   31  016D   245          brw     fail                            ;and fail
```

```
                20 AE    0A   D1  0170   246 20$:    cmpl    #dat_k_char,32(sp)          ;were we converting to char?
                   F0    13   0174   247          beql    15$                         ;if eql, yes, it failed once, so no
                         0176   248                                                     ;onkey value will be returned
                20 AE    0B   D0  0176   249          movl    #dat_k_char_var,32(sp)      ;set for char var dest
          24 AE   00000100 8F   D0  017A   250          movl    #256,36(sp)                 ;set for length of 256
             5E   00000102 8F   C2  0182   251          subl    #258,sp                     ;get room for char temp
                  012E CE    5E   D0  0189   252          movl    sp,258+44(sp)               ;set addr of dest
       00000000'GF    010A CE    FA  018E   253          callg   258+8(sp),g^pli$cvrt_any    ;convert it to char var
                         5E   DD  0197   254          pushl   sp                          ;set addr of onkey value
                         52   DD  0199   255          pushl   r2                          ;set fcb address
                   00000000'8F   DD  019B   256          pushl   #pli$_cnverr                ;set conversion error
                   00000000'8F   DD  01A1   257          pushl   #pli$_key                   ;set key condition
              00000000'GF    04   FB  01A7   258          calls   #4,g^pli$io_error           ;signal condition
                         04   01AE   259          ret                                 ;return
                              01AF   260
                              01AF   261 ;++
                              01AF   262 ;pli$$chk_keycnd
                              01AF   263 ; this routine checks the rms rab for status that should be signaled as
                              01AF   264 ; the pl1 key condition. if such a status is found, the onkey value is
                              01AF   265 ; calculated, and io_error is called to signal the key condition.
                              01AF   266 ; inputs:
                              01AF   267 ;       r0 = error sub-code
                              01AF   268 ;       r2 = fcb address
                              01AF   269 ;       r3 = address of key descriptor
                              01AF   270 ;       r4 = rab address
                              01AF   271 ; outputs:
                              01AF   272 ;       none
                              01AF   273 ;--
                              01AF   274
                              01AF   275 pli$$chk_keycnd::
          50    00000000'8F   D1  01AF   276          cmpl    #pli$_rmsr,r0               ;rab error?
                         01   13   01B6   277          beql    20$                         ;if eql, yes, cont
                         05   01B8   278 10$:     rsb                                 ;return
                         53   D5   01B9   279 20$:     tstl    r3                          ;key specified?
                         FB   13   01BB   280          beql    10$                         ;if eql, no, just return
                         63   D5   01BD   281          tstl    (r3)                        ;key addr specified?
                         F7   13   01BF   282          beql    10$                         ;if eql, no, just return
       08 A4    00000000'8F   D1  01C1   283          cmpl    #rms$_rnf,rab$l_sts(r4)     ;record not found?
                         28   13   01C9   284          beql    30$                         ;if eql, yes, raise key
       08 A4    00000000'8F   D1  01CB   285          cmpl    #rms$_key,rab$l_sts(r4)     ;key error?
                         1E   13   01D3   286          beql    30$                         ;if eql, yes, raise key
       08 A4    00000000'8F   D1  01D5   287          cmpl    #rms$_mrn,rab$l_sts(r4)     ;max rec num exceeded?
                         14   13   01DD   288          beql    30$                         ;if eql, yes, raise key
       08 A4    00000000'8F   D1  01DF   289          cmpl    #rms$_dup,rab$l_sts(r4)     ;duplicate key?
                         0A   13   01E7   290          beql    30$                         ;if eql, yes, raise key
       08 A4    00000000'8F   D1  01E9   291          cmpl    #rms$_rex,rab$l_sts(r4)     ;record already exists?
                         C5   12   01F1   292          bneq    10$                         ;if neq, no, just return
          5E    00000102 8F   C2  01F3   293 30$:     subl    #258,sp                     ;get room for vcha temp
                         5E   DD  01FA   294          pushl   sp                          ;set addr of dest
                         52   DD  01FC   295          pushl   r2                          ;set fcb addr
                         50   DD  01FE   296          pushl   r0                          ;set error code
                   00000000'8F   DD  0200   297          pushl   #pli$_key                   ;set key condition
                         00   DD  0206   298          pushl   #0                          ;set dest offset
                   00000100 8F   DD  0208   299          pushl   #256                        ;set dest size
                         0B   DD  020E   300          pushl   #dat_k_char_var             ;set dest data type
                      18 AE   9F  0210   301          pushab  24(sp)                      ;set addr of addr of dest
                         00   DD  0213   302          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   360           movab   40(sp),sp                       ;clean stack
          59 00BC  C2    05    E0   0287   361           bbs     #fab$v_bio,<fcb_b_fab+fab$b_fac>(r2),300$ ;if blockio, cont
             46  0C  A2    10    E1   028D   362           bbc     #atr_v_indexed,fcb_l_attr(r2),200$ ;if indexed
                     65    0A    D1   0292   363           cmpl    #dat_k_char,kcb_l_dtyp(r5) ;character key?
                           28    12   0295   364           bneq    160$                            ;if neq, no
                           1C    BB   0297   365           pushr   #^m<r2,r3,r4>                   ;save regs
                   51      0C AE    9E   0299   366           movab   12(sp),r1                       ;get start addr of key
                   50      04 A5    D0   029D   367           movl    kcb_l_prec(r5),r0               ;get total length of key
                   34 A4           50    90   02A1   368           movb    r0,rab$b_ksz(r4)                ;set size in rab
                   57      28 A4    D0   02A5   369           movl    rab$l_rbf(r4),r7                ;get addr of buffer
                   58      08 A5    9E   02A9   370           movab   kcb_w_pos0(r5),r8               ;get addr of pos0 in kcb
                           53    B8   3C   02AD   371 120$:     movzwl  (r8)+,r3                        ;get position in record
                           53    57   C0   02B0   372           addl    r7,r3                           ;add addr of buffer
      63   88   20     61    50   2C   02B3   373           movc5   r0,(r1),#^x20,(r8)+,(r3)        ;copy this segment
                           F2    1A   02B9   374           bgtru   120$                            ;if gtru, more to do, go again
                           1C    BA   02BB   375           popr    #^m<r2,r3,r4>                   ;restore regs
                           1D    11   02BD   376           brb     210$                            ;cont
                   51      08 A5    3C   02BF   377 160$:     movzwl  kcb_w_pos0(r5),r1               ;get position in buffer
                   51      28 A4    C0   02C3   378           addl    rab$l_rbf(r4),r1                ;add in addr of buffer
                           1C    BB   02C7   379           pushr   #^m<r2,r3,r4>                   ;save regs
                34 A4      0A A5    90   02C9   380           movb    kcb_w_len0(r5),rab$b_ksz(r4) ;set size of key in rab
             61    0C AE    0A A5    28   02CE   381           movc3   kcb_w_len0(r5),12(sp),(r1) ;copy key to buffer
                           1C    BA   02D4   382           popr    #^m<r2,r3,r4>                   ;restore regs
                           04    11   02D6   383           brb     210$                            ;cont
                34 A4      04    90   02D8   384 200$:     movb    #4,rab$b_ksz(r4)                ;set size of key in rab
                30 A4      5E    D0   02DC   385 210$:     movl    sp,rab$l_kbf(r4)                ;set key buffer addr in rab
                1E A4      01    90   02E0   386           movb    #rab$c_key,rab$b_rac(r4) ;set keyed access in rab
                           66    17   02E4   387           jmp     (r6)                            ;return
                     38 A4 8ED0    02E6   388 300$:     popl    rab$l_bkt(r4)                   ;set lbn for block io
                           66    17   02EA   389           jmp     (r6)                            ;return
                                      02EC   390           .enabl  lsb
                                      02EC   391
                                      02EC   392 ;++
                                      02EC   393 ;pli$$keynum
                                      02EC   394 ; inputs
                                      02EC   395 ;       r0 - addr of keynum option
                                      02EC   396 ;       r1 - addr of key option
                                      02EC   397 ;       r2 - addr of fcb
                                      02EC   398 ;       r3 - fcb_l_attr
                                      02EC   399 ;       r4 - addr of rab
                                      02EC   400 ; outputs
                                      02EC   401 ;       r1 - addr of key option
                                      02EC   402 ;--
                                      02EC   403
                                      02EC   404 pli$$keynum::
                         50    D5   02EC   405           tstl    r0                              ;keynum passed?
                         56    13   02EE   406           beql    90$                             ;if eql, no, just return
                   0A 53       10   02F0   407           bbs     #atr_v_indexed,r3,20$           ;if indexed, cont
          50   00000000'8F    D0   02F4   408 10$:      movl    #pli$_notindexed,r0             ;set not indexed file
                       0157    31   02FB   409           brw     fail                            ;and fail
                   0A 53       08   02FE   410 20$:      bbs     #atr_v_keyed,r3,40$             ;if keyed, cont
          50   00000000'8F    D0   0302   411 30$:      movl    #pli$_notkeyd,r0                ;set not keyed file
                       0149    31   0309   412           brw     fail                            ;and fail
           0A 00BC  C2    05    E1   030C   413 40$:      bbc     #fab$v_bio,<fcb_b_fab+fab$b_fac>(r2),45$ ;if not blockio, cont
          50   00000000'8F    D0   0312   414 42$:      movl    #pli$_conblokio,r0              ;set conflicting block io
                       0139    31   0319   415           brw     fail                            ;and fail
                         51    D5   031C   416 45$:      tstl    r1                              ;key specified?
```

```
                 0A   12  031E   417                bneq    60$                             ;if neq, yes, cont
      50 00000000'8F   D0  0320   418 50$:          movl    #pli$_nokey,r0                  ;set no key specified
            012B   31  0327   419                brw     fail                            ;and fail
         50   60   3C  032A   420 60$:          movzwl  (r0),r0                         ;get key number
                 0A   18  032D   421                bgeq    80$                             ;if geq, cont
      50 00000000'8F   D0  032F   422 70$:          movl    #pli$_invindnum,r0              ;set invalid key number
            011C   31  0336   423                brw     fail                            ;and fail
      7E    3C A2   9A  0339   424 80$:          movzbl  fcb_b_numkcbs(r2),-(sp) ;get highest key number + 1
         8E   50   D1  033D   425                cmpl    r0,(sp)+                        ;key num too big?
              ED   18  0340   426                bgeq    70$                             ;if geq, then yes, fail
      35 A4   50   90  0342   427                movb    r0,rab$b_krf(r4)                ;set key of ref in rab
              05   05  0346   428 90$:          rsb
                   0347   429
                   0347   430 ;++
                   0347   431 ;pli$$matchgtr
                   0347   432 ; inputs:
                   0347   433 ;       r0 - addr of match_greater
                   0347   434 ;       r1 - addr of key
                   0347   435 ;       r2 - addr of fcb
                   0347   436 ;       r3 - fcb_l_attr
                   0347   437 ;       r4 - addr of rab
                   0347   438 ; outputs:
                   0347   439 ;       r1 - addr of key
                   0347   440 ;--
                   0347   441
                   0347   442 pli$$matchgtr::
           16   DD  0347   443                pushl   #rab$v_kgt                      ;set field index of key greater
           02   11  0349   444                brb     100$                            ;cont in common
                   034B   445
                   034B   446 ;++
                   034B   447 ;pli$$matchgeq
                   034B   448 ; inputs:
                   034B   449 ;       r0 - addr of match_greater_eql
                   034B   450 ;       r1 - addr of key
                   034B   451 ;       r2 - addr of fcb
                   034B   452 ;       r3 - fcb_l_attr
                   034B   453 ;       r4 - addr of rab
                   034B   454 ; outputs
                   034B   455 ;       r1 - addr of key
                   034B   456 ;--
                   034B   457
                   034B   458 pli$$matchgeq::
           15   DD  034B   459                pushl   #rab$v_kge                      ;set field index of key greater
                   034D   460 100$:
           50   D5  034D   461                tstl    r0                              ;match gtr specified?
           14   13  034F   462                beql    110$                            ;if eql, no, just return
         AD 53  08   E1  0351   463                bbc     #atr_v_keyed,r3,30$     ;if not keyed file, fail
      B7 00BC C2   05   E0  0355   464                bbs     #fab$v_bio,<fcb_b_fab+fab$b_fac>(r2),42$ ;if blockio, fail
           51   D5  035B   465                tstl    r1                              ;key specified?
           C1   13  035D   466                beql    50$                             ;if eql, no, fail
04 A4   01   6E  60   F0  035F   467                insv    (r0),(sp),#1,rab$l_rop(r4) ;set match greater or greatereql
         5E   04 AE   9E  0365   468 110$:          movab   4(sp),sp                        ;remove index
              05   05  0369   469                rsb                                     ;return
                   036A   470
                   036A   471 ;++
                   036A   472 ;pli$$valrecidto
                   036A   473 ; inputs:
```

```
                     036A   474 ;       r0 - addr of record id to
                     036A   475 ;       r2 - addr of fcb
                     036A   476 ;       r3 - fcb_l_attr
                     036A   477 ;       r4 - addr of rab
                     036A   478 ;--
                     036A   479
                     036A   480 pli$$valrecidto::
              50  D5 036A   481           tstl    r0                          ;record id to specified?
              0E  13 036C   482           beql    120$                        ;if eql, no, just return
        0A 53 0F  E0 036E   483           bbs     #atr_v_recidacc,r3,120$ ;if rec id, cont
    50 00000000'8F  D0 0372   484 115$:   movl    #pli$_recid,r0              ;set not record id file
            00D9  31 0379   485           brw     fail                        ;and fail
              05  03A0   486 120$:   rsb                                     ;return
                  03A1   487
                  03A1   488 ;++
                  03A1   489 ;pli$$recidfrom
                  03A1   490 ; inputs:
                  03A1   491 ;       r0 - addr of record id from
                  03A1   492 ;       r1 - addr of key
                  03A1   493 ;       r2 - addr of fcb
                  03A1   494 ;       r3 - fcb_l_attr
                  03A1   495 ;       r4 - addr of rab
                  03A1   496 ; outputs:
                  03A1   497 ;       condition code z bit is clear if rfa from specified
                  03A1   498 ;--
                  03A1   499
                  03A1   500 pli$$recidfrom::
              50  D5 03A1   501           tstl    r0                          ;record id from specified?
              FB  13 03A3   502           beql    120$                        ;if eql, no, just return
        ED 53 0F  E1 03A5   503           bbc     #atr_v_recidacc,r3,115$ ;if not recid, fail
              51  D5 03A9   504           tstl    r1                          ;key also specified?
              0A  13 03AB   505           beql    130$                        ;if eql, no, cont
    50 00000000'8F  D0 03AD   506           movl    #pli$_recidkey,r0           ;set record id and key conflict
            00C2  31 03AE   507           brw     fail                        ;and fail
        10 A4 60  7D 03B1   508 130$:   movq    (r0),rab$w_rfa(r4)          ;set recid in rab
        1E A4 02  90 03B5   509           movb    #rab$c_rfa,rab$b_rac(r4) ;set for rfa access in rab
           35 A4  94 03B9   510           clrb    rab$b_krf(r4)               ;reset key of ref in rab
              04  B9 03BC   511           bicpsw  #4                          ;clear condition code Z bit
              05  03BE   512           rsb                                 ;return
                  03BF   513
                  03BF   514 ;++
                  03BF   515 ;pli$$fxdctlfrom
                  03BF   516 ; inputs:
                  03BF   517 ;       r0 - addr of fixed control from descr. 0(r0) is the addr, 4(r0)
                  03BF   518 ;                 is word size/prec, 6(r0) is word data type
                  03BF   519 ;       r2 - addr of fcb
                  03BF   520 ;       r3 - fcb_l_attr
                  03BF   521 ;       r4 - addr of rab
                  03BF   522 ;--
                  03BF   523
                  03BF   524 pli$$fxdctlfrom::
           2C A4  D4 03BF   525           clrl    rab$l_rhb(r4)               ;assume not specified
              60  DD 03C2   526           pushl   (r0)                        ;save addr of fixed ctl
              45  13 03C4   527           beql    150$                        ;if eql, none specified
  0A 00BC C2  05  E1 03C6   528           bbc     #fab$v_bio,<fcb_b_fab+fab$b_fac>(r2),134$ ;if not blockio, cont
    50 00000000'8F  D0 03CC   529 133$:   movl    #pli$_conblokio,r0          ;set conflicts with blockio
            009D  31 03D3   530           brw     fail                        ;and fail
```

```
                06 A0    0B   B1  03B8   531 134$:   cmpw    #dat_k_char_var,6(r0)   ;is it char var?
                   06    12   3C  03BC   532         bneq    135$                    ;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$$bytesize      ;get size
                15 50       E9  03CE   537         blbc    r0,139$                 ;if lbc, fail
                   51       D5  03D1   538 137$:   tstl    r1                      ;anything there?
                   18       15  03D3   539         bleq    150$                    ;if leq, no, just return
             50 00E5 C2     9A  03D5   540         movzbl  <fcb_b_fab+fab$b_fsz>(r2),r0 ;get fixed control size
                51 50       D1  03DA   541         cmpl    r0,r1                   ;enuf room?
                   09       13  03DD   542         beql    140$                    ;if geq, yes, cont
          50 00000000'8F    D0  03DF   543         movl    #pli$_fxcsiz,r0         ;set fixed control wrong size
                   6[       11  03E6   544 139$:   brb     fail                    ;and fail
                2C [ , 8ED0  03E8   545 140$:   popl    rab$l_rhb(r4)           ;set addr in rab
                   05       03EC   546         rsb                             ;return
                5E 04 AE    9E  03ED   547 150$:   movab   4(sp),sp                ;clean stack
                   05       03F1   548         rsb                             ;return
                            03F2   549
                            03F2   550 ;++
                            03F2   551 ;pli$$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       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$l_rhb(r4)      ;set addr in rab
                05   04  0454   593 160$:     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
ATR_V_KEYED             =  00000008
ATR_V_RECIDACC          =  0000000F
DAT_K_CHAR              =  0000000A
DAT_K_CHAR_VAR          =  0000000B
DAT_K_FIX_BIN           =  00000002
DAT_K_STRUCTURE         =  00000017
FAB$B_FAC               =  00000016
FAB$B_FSZ               =  0000003F
FAB$V_BIO               =  00000005
FAIL                    00000455 R      02
FCB_B_ENVIR             000001C2
FCB_B_ESA               0000012E
FCB_B_EXTRA             0000003D
FCB_B_FAB               000000A6
FCB_B_IDENT             00000040
FCB_B_IDENT_NAM         00000042
FCB_B_NAM               000000F6
FCB_B_NUMKCBS           0000003C
FCB_B_RAB               00000062
FCB_C_LEN               000001C2
FCB_C_STRLEN            00000034
FCB_L_ATTR              0000000C
FCB_L_BUF               00000014
FCB_L_BUF_END           00000018
FCB_L_BUF_PT            0000001C
FCB_L_CNDADDR           000001B2
FCB_L_CONDIT            000001AE
FCB_L_DTTR              00000010
FCB_L_ERROR             00000008
FCB_L_KCB               00000038
FCB_L_NEXT              00000000
FCB_L_PREVIOUS          00000004
FCB_L_PRN               00000034
FCB_Q_RFA               00000020
FCB_W_COLUMN            0000002E
FCB_W_IDENT_LEN         00000040
FCB_W_LINE              00000030
FCB_W_LINESIZE          0000002A
FCB_W_PAGE              00000032
FCB_W_PAGESIZE          0000002C
FCB_W_REVISION          00000028
KCB_C_LEN               0000002C
KCB_L_DTYP              00000000
KCB_L_PREC              00000004
KCB_L_ZERO              00000028
KCB_W_LEN0              0000000A
KCB_W_LEN1              0000000E
KCB_W_LEN2              00000012
KCB_W_LEN3              00000016
KCB_W_LEN4              0000001A
KCB_W_LEN5              0000001E
KCB_W_LEN6              00000022
KCB_W_LEN7              00000026
KCB_W_POS0              00000008
KCB_W_POS1              0000000C
KCB_W_POS2              00000010
KCB_W_POS3              00000014
KCB_W_POS4              00000018
KCB_W_POS5              0000001C
KCB_W_POS6              00000020
KCB_W_POS7              00000024
KEY_CVRT                00000140 R      02
PLI$$BYTESIZE           ********   X    02
PLI$$CHK_KEYCND         000001AF RG     02
PLI$$FXCTLTO_R6         000003F2 RG     02
PLI$$FXDCTLFROM         000003A1 RG     02
PLI$$KEYNUM             000002EC RG     02
PLI$$KEYTO_R8           00000000 RG     02
PLI$$KEY_HND            00000085 RG     02
PLI$$MATCHGEQ           0000034B RG     02
PLI$$MATCHGTR           00000347 RG     02
PLI$$READKEY_R6         000000C2 RG     02
PLI$$RECIDFROM          0000037D RG     02
PLI$$VALRECIDTO         0000036A RG     02
PLI$$WRITEKEY_R8        0000022C RG     02
PLI$CVRT_ANY            ********    X   02
PLI$GOTO                ********    X   02
PLI$IO_ERROR            ********    X   02
PLI$_CNVERR             ********    X   02
PLI$_CONBLOKIO          ********    X   02
PLI$_ERROR              ********    X   02
PLI$_FXCSIZ             ********    X   02
PLI$_INVINDNUM          ********    X   02
PLI$_KEY                ********    X   02
PLI$_NOKEY              ********    X   02
PLI$_NOTINDEXED         ********    X   02
PLI$_NOTKEYD            ********    X   02
PLI$_RECID              ********    X   02
PLI$_RECIDKEY           ********    X   02
PLI$_RMSR               ********    X   02
RAB$B_KRF               =  00000035
RAB$B_KSZ               =  00000034
RAB$B_RAC               =  0000001E
RAB$C_KEY               =  00000001
RAB$C_RFA               =  00000002
RAB$L_BKT               =  00000038
RAB$L_KBF               =  00000030
RAB$L_RBF               =  00000028
RAB$L_RHB               =  0000002C
RAB$L_ROP               =  00000004
RAB$L_STS               =  00000008
RAB$L_UBF               =  00000024
RAB$V_KGE               =  00000015
RAB$V_KGT               =  00000016
RAB$W_RFA               =  00000010
RMS$_DUP                ********    X   02
RMS$_KEY                ********    X   02
RMS$_MRN                ********    X   02
RMS$_REX                ********    X   02
RMS$_RNF                ********    X   02
SIGERR                  00000073 R      02
SIZ...                  =  00000001
SS$_RESIGNAL            ********    X   02
```

```
STK_L_AP                        00000008
STK_L_ARG_LIST                  FFFFFFF8
STK_L_CND_HND                   00000000
STK_L_CND_LST                   FFFFFFF4
STK_L_DISPLAY                   FFFFFFFC
STK_L_FP                        0000000C
STK_L_PC                        00000010
STK_L_PSL                       00000004
STK_L_REGS                      00000014
```

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

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