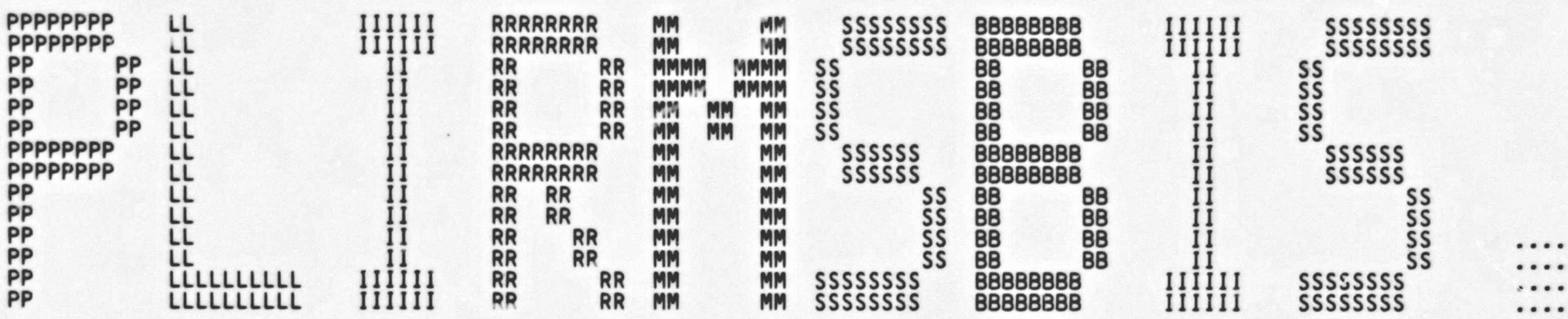
```
PPPPPPPPPPPP      LLL                IIIIIIIII    RRRRRRRRRRRR      TTTTTTTTTTTTTTT   LLL
PPPPPPPPPPPPP     LLL                IIIIIIIII    RRRRRRRRRRRRR     TTTTTTTTTTTTTTT   LLL
PPPPPPPPPPPPP     LLL                IIIIIIIII    RRRRRRRRRRRRR     TTTTTTTTTTTTTTT   LLL
PPP          PPP  LLL                   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**PLIRMSBIS



```
0000     1          .title  pli$rmsbis - pl/i rms built-in subroutines
0000     2          .ident  /1-003/                                 ; Edit BLS0294
0000     3 ;
0000     4 ;*****************************************************************************
0000     5 ;*                                                                           *
0000     6 ;*  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
0000     7 ;*  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
0000     8 ;*  ALL RIGHTS RESERVED.                                                     *
0000     9 ;*                                                                           *
0000    10 ;*  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
0000    11 ;*  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE   *
0000    12 ;*  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
0000    13 ;*  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
0000    14 ;*  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
0000    15 ;*  TRANSFERRED.                                                             *
0000    16 ;*                                                                           *
0000    17 ;*  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
0000    18 ;*  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT   *
0000    19 ;*  CORPORATION.                                                             *
0000    20 ;*                                                                           *
0000    21 ;*  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
0000    22 ;*  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
0000    23 ;*                                                                           *
0000    24 ;*                                                                           *
0000    25 ;*****************************************************************************
0000    26 ;
0000    27 ;++
0000    28 ; facility:
0000    29 ;
0000    30 ;       VAX/VMS PL1 runtime library
0000    31 ;
0000    32 ; abstract:
0000    33 ;
0000    34 ;       This module contains the pl1 runtime routines for the RMS built-in
0000    35 ;       subroutines.
0000    36 ;
0000    37 ; author: c. spitz 8-may-1980
0000    38 ;
0000    39 ; Modifications:
0000    40 ;
0000    41 ;
0000    42 ;       1-002   Bill Matthews   29-September-1982
0000    43 ;
0000    44 ;               Invoke macros $defdat and rtshare instead of $defopr and share.
0000    45 ;
0000    46 ;       1-003   Benn Schreiber   6-APR-1984
0000    47 ;               Zero XAB's before use.
0000    48 ;--
0000    49
0000    50 ;
0000    51 ; external definitions
0000    52 ;
0000    53          $deffcb                         ;define file control block
0000    54          $rabdef                         ;define rab offsets
0000    55          $fabdef                         ;define fab offsets
0000    56          $namdef                         ;define nam offsets
0000    57          $devdef                         ;define device bit offsets
```

```
0000     58          $defdsp                         ;define file display block
0000     59          $defdat                         ;define operand node data types
0000     60          $xabdef                         ;define xab offsets
0000     61          $xabprodef                      ;define protection xab
0000     62          $xabdatdef                      ;define date xab
0000     63          $xabfhcdef                      ;define file header char xab
0000     64
0000     65          rtshare
0000     66
```

```
                                0000     68 ;++
                                0000     69 ; pli$display - display file attributes
                                0000     70 ; this routine displays the attributes of a pl1 file. if the file is closed,
                                0000     71 ; it is opened with declare time attributes. the users display block is then
                                0000     72 ; filled in.
                                0000     73 ;
                                0000     74 ; inputs:
                                0000     75 ;       0(ap) - number of arguments = 3
                                0000     76 ;       4(ap) - address of file control block
                                0000     77 ;       8(ap) - address of users display block
                                0000     78 ;       12(ap) - size/prec of display block
                                0000     79 ;        14(ap) - data type of display block
                                0000     80 ; outputs:
                                0000     81 ;       the users display block is filled in with the files current attributes.
                                0000     82 ;--
                            0FFC 0000     83          .entry  pli$display,^m<r2,r3,r4,r5,r6,r7,r8,r9,r10,r11>
                    56    04 AC   D0 0002     84          movl    4(ap),r6                ;get address of fcb
                    57    08 AC   D0 0006     85          movl    8(ap),r7                ;get address of dsp
                          0C AC   DD 000A     86          pushl   12(ap)                  ;push size and data type
            00000000'GF     01   FB 000D     87          calls   #1,g^pli$$bytesize      ;calculate byte size
                       1A   50   E9 0014     88          blbc    r0,5$                   ;if invalid data type, fail
            00000121 8F     51   D1 0017     89          cmpl    r1,#dspend              ;user block large enough?
                            11   19 001E     90          blss    5$                      ;if lss, no, fail
                    0B    0E AC   B1 0020     91          cmpw    14(ap),#dat_k_char_var  ;char var block?
                            18   12 0024     92          bneq    7$                      ;if neq, no, cont
                87    0121 8F   B0 0026     93          movw    #dspend,(r7)+           ;set size for char var
                    08 AC   57   D0 002B     94          movl    r7,8(ap)                ;set correct address of display
                            0D   11 002F     95          brb     7$                      ;cont
            50    00000000'8F   D0 0031     96 5$:      movl    #pli$_invdatyp,r0       ;set invalid data type
                         52   56   D0 0038     97          movl    r6,r2                   ;set fcb addr
                            0357 31 003B     98          brw     fail                    ;and fail
                      58    00A6 C6 003E     99 7$:      movab   fcb_b_fab(r6),r8        ;get address of fab
                        59    62 A6 0043    100          movab   fcb_b_rab(r6),r9        ;get address of rab
                      55    00F6 C6 0047    101          movab   fcb_b_nam(r6),r5        ;get address of nam
                         5B   57   D0 004C    102          movl    r7,r11                  ;copy dsp address
                   1D 0C A6   01   E0 004F    103          bbs     #atr_v_opened,fcb_l_attr(r6),10$ ;if file opened, cont
                            00   DD 0054    104          pushl   #0                      ;set no implied attributes
                            56   DD 0056    105          pushl   r6                      ;set fcb addr
            00000000'GF     02   FB 0058    106          calls   #2,g^pli$open           ;open the file
                   0D 0C A6   01   E0 005F    107          bbs     #atr_v_opened,fcb_l_attr(r6),10$ ;if file still not opened
            50    00000000'8F   D0 0064    108          movl    #pli$_open,r0           ;set open failure
                         52   56   D0 006B    109          movl    r6,r2                   ;copy fcb addr
                            0324 31 006E    110          brw     fail                    ;and fail
                         5E   2C   C2 0071    111 10$:    subl    #xab$k_datlen,sp        ;get room for date xab
                         50   5E   D0 0074    112          movl    sp,r0                   ;save xab address
                            3C   BB 0077    113          pushr   #^m<r2,r3,r4,r5>        ;save registers
        60    2C    00    6E    00 2C 0079    114          movc5   #0,(sp),#0,#xab$k_datlen,(r0) ;zero xab
                            3C   BA 007F    115          popr    #^m<r2,r3,r4,r5>
                         6E   12   90 0081    116          movb    #xab$c_dat,xab$b_cod(sp) ;set xab type
                      01 AE   2C   90 0084    117          movb    #xab$k_datlen,xab$b_bln(sp) ;set xab length
                         52   5E   D0 0088    118          movl    sp,r2                   ;save addr of date xab
                      24 A8   52   D0 008B    119          movl    r2,fab$l_xab(r8)        ;link date xab to fab
                   5E    00000058 8F   C2 008F    120          subl    #xab$k_prolen,sp        ;get room for protection xab
                         50   5E   D0 0096    121          movl    sp,r0                   ;save xab address
                            3C   BB 0099    122          pushr   #^m<r2,r3,r4,r5>        ;save registers
    60    0058 8F    00    6E    00 2C 009B    123          movc5   #0,(sp),#0,#xab$k_prolen,(r0) ;zero xab
                            3C   BA 00A3    124          popr    #^m<r2,r3,r4,r5>
```

```
                6E   13    90   00A5   125          movb    #xab$c_pro,xab$b_cod(sp) ;set xab type
             01 AE   58 8F   90   00A8   126          movb    #xab$k_prolen,xab$b_bln(sp) ;set xab length
                5A    5E    D0   00AD   127          movl    sp,r10                   ;save addr of protection xab
             04 A2    5A    D0   00B0   128          movl    r10,xab$l_nxt(r2)        ;link protection xab to date xab
                5E    2C    C2   00B4   129          subl    #xab$k_fhclen,sp         ;get room for file header char xab
                50    5E    D0   00B7   130          movl    sp,r0                    ;save xab address
                      3C    BB   00BA   131          pushr   #^m<r2,r3,r4,r5>         ;save registers
60   2C   00    6E    00    2C   00BC   132          movc5   #0,(sp),#0,#xab$k_fhclen,(r0) ;zero xab
                      3C    BA   00C2   133          popr    #^m<r2,r3,r4,r5>
                6E    1D    90   00C4   134          movb    #xab$c_fhc,xab$b_cod(sp) ;set xab type
             01 AE    2C    90   00C7   135          movb    #xab$k_fhclen,xab$b_bln(sp) ;set xab length
             04 AA    5E    D0   00CB   136          movl    sp,xab$l_nxt(r10)        ;link fhc xab to protection xab
                04 AE    D4   00CF   137          clrl    xab$l_nxt(sp)            ;clr next xab link
                         00D2   138          $display (r8)                    ;get the info
                24 A8    D4   00DB   139          clrl    fab$l_xab(r8)            ;unlink xab's from fab
                0D 50    E8   00DE   140          blbs    r0,15$                   ;if lbs, cont
          50    00000000'8F    D0   00E1   141          movl    #pli$_rmsf,r0            ;set fab error
                52    56    D0   00E8   142          movl    r6,r2                    ;set fcb addr
                   02A7    31   00EB   143          brw     fail                     ;and fail
             8B    3C A8    3C   00EE   144 15$:     movzwl  fab$w_bls(r8),(r11)+     ;set block size
             8B    3E A8    9A   00F2   145          movzbl  fab$b_bks(r8),(r11)+     ;set bucket size
             8B    14 A2    7D   00F6   146          movq    xab$q_cdt(r2),(r11)+     ;set creation date
             8B    1C A2    7D   00FA   147          movq    xab$q_edt(r2),(r11)+     ;set expiration date
             8B    14 A8    3C   00FE   148          movzwl  fab$w_deq(r8),(r11)+     ;set extension size
          6B    14 A5    16    28   0102   149          movc3   #22,nam$t_dvi(r5),(r11) ;set file id and dvi
             5B    18 AB    9E   0107   150          movab   24(r11),r11              ;update pointer in dsp
             8B    0C AE    D0   010B   151          movl    xab$l_hbk(sp),(r11)+     ;set file size
             8B    3F A8    9A   010F   152          movzbl  fab$b_fsz(r8),(r11)+     ;set fixed control size
             8B    35 A9    9A   0113   153          movzbl  rab$b_krf(r9),(r11)+     ;set index number
             8B    38 A8    D0   0117   154          movl    fab$l_mrn(r8),(r11)+     ;set maximum record number
             8B    36 A8    3C   011B   155          movzwl  fab$w_mrs(r8),(r11)+     ;set maximum record size
             8B    37 A9    9A   011F   156          movzbl  rab$b_mbc(r9),(r11)+     ;set multiblock count
             8B    36 A9    9A   0123   157          movzbl  rab$b_mbf(r9),(r11)+     ;set multibuffer count
             8B    0E AA    3C   0127   158          movzwl  xab$w_grp(r10),(r11)+    ;set owner group
             8B    0C AA    3C   012B   159          movzwl  xab$w_mbm(r10),(r11)+    ;set owner member
             8B    1C A8    9A   012F   160          movzbl  fab$b_rtv(r8),(r11)+     ;set retrieval pointers
             8B    2A A6    3C   0133   161          movzwl  fcb_w_linesize(r6),(r11)+ ;set linesize
             8B    2C A6    3C   0137   162          movzwl  fcb_w_pagesize(r6),(r11)+ ;set pagesize
             8B    32 A6    3C   013B   163          movzwl  fcb_w_page(r6),(r11)+    ;set page number
             8B    30 A6    3C   013F   164          movzwl  fcb_w_line(r6),(r11)+    ;set line number
             8B    2E A6    3C   0143   165          movzwl  fcb_w_column(r6),(r11)+ ;set column number
             8B    3C A6    9A   0147   166          movzbl  fcb_b_numkcbs(r6),(r11)+ ;set number of keys
                   6B    7C   014B   167          clrq    (r11)                    ;clear all of dtr
                         CB   014D   168          bicl3   #^c<fab$m_sup!fab$m_tmp! - ;get supersede, temporary
                         014E   169                  fab$m_dfw!fab$m_rwo! -   ;deferred_write, rewind_on_open
                         014E   170                  fab$m_pos!fab$m_wck! -   ;current_position, write_check
                         014E   171                  fab$m_rwc!fab$m_spl! -   ;rewind_on_close, spool
                         014E   172                  fab$m_scf!fab$m_dlt! -   ;batch, delete
                         014E   173                  fab$m_ctg!fab$m_cbt! -   ;contiguous, contiguous_best_try
                         014E   174                  fab$m_rck!fab$m_tef>,-   ;read_check and truncate bits
6B   04 A8    EF4F1453 8F         014E   175                  fab$l_fop(r8),(r11)      ;from fop and set in display
                0D    04   EF   0156   176          extzv   #atr_v_update,#<atr_v_indexed+1-atr_v_update>,- ;get bits from
             50    0C A6         0159   177                  fcb_l_attr(r6),r0        ;fcb attributes
                1D    50   F0   015C   178          insv    r0,#dtr_v_update,-       ;insert them
                6B    0D         015F   179                  #<dtr_v_indexed+1-dtr_v_update>,(r11) ;in display
             51    1E A8    9A   0161   180          movzbl  fab$b_rat(r8),r1         ;get rat from fab
          50   51    01    00   EF   0165   181          extzv   #fab$v_ftn,#1,r1,r0      ;get ftn from rat
```

```
                6B   01   00   50   F0  016A   182          insv    r0,#dtr_v_fortran_format,#1,(r11) ;set fortran format
                50   51   01   01   EF  016F   183          extzv   #fab$v_cr,#1,r1,r0          ;get cr from rat
                6B   01   06   50   F0  0174   184          insv    r0,#dtr_v_carriage_return_format,#1,(r11) ;set carriage return
                50   51   01   02   EF  0179   185          extzv   #fab$v_prn,#1,r1,r0         ;get prn from rat
                6B   01   12   50   F0  017E   186          insv    r0,#dtr_v_printer_format,#1,(r11) ;set printer format
                50   51   01   03   EF  0183   187          extzv   #fab$v_blk,#1,r1,r0         ;get blk from rat
                6B   01   01   50   F0  0188   188          insv    r0,#dtr_v_block_boundary_format,#1,(r11) ;set block boundary
           50     16 A8   01   05   EF  018D   189          extzv   #fab$v_bio,#1,fab$b_fac(r8),r0 ;get bio from fac
                6B   01   04   50   F0  0193   190          insv    r0,#dtr_v_block_io,#1,(r11) ;set block io
                      1F A8   01   91  0198   191          cmpb    #fab$c_fix,fab$b_rfm(r8) ;fixed length records?
                              07   12  019C   192          bneq    16$                         ;if neq, no, cont
                 6B   00000400 8F   C8  019E   193          bisl    #dtr_m_fixed_length_records,(r11) ;set fixed length records
           50     0C A6   01   15   EF  01A5   194 16$:     extzv   #atr_v_app_comma,#1,fcb_l_attr(r6),r0 ;get append comma
                          50   50   92  01AB   195          mcomb   r0,r0                       ;compliment it
                6B   01   0C   50   F0  01AE   196          insv    r0,#dtr_v_ignore_line_marks,#1,(r11) ;set ignore linemarks
                        51   04   A9  01B3   197          movl    rab$l_rop(r9),r1            ;get rop from rab
                50   51   01   0D   EF  01B7   198          extzv   #rab$v_loa,#1,r1,r0         ;get loa from rop
                6B   01   10   50   F0  01BC   199          insv    r0,#dtr_v_initial_fill,#1,(r11) ;set initial fill
                50   51   01   09   EF  01C1   200          extzv   #rab$v_rah,#1,r1,r0         ;get rah from rop
                6B   01   16   50   F0  01C6   201          insv    r0,#dtr_v_read_ahead,#1,(r11) ;set read ahead
                50   51   01   0A   EF  01CB   202          extzv   #rab$v_wbh,#1,r1,r0         ;get wbh from rop
                6B   01   13   50   F0  01D0   203          insv    r0,#dtr_v_write_behind,#1,(r11) ;set write behind
                        51   17   A8  90  01D5   204          movb    fab$b_shr(r8),r1            ;get shr from fab
                50   51   01   05   EF  01D9   205          extzv   #fab$v_nil,#1,r1,r0         ;get nil from shr
                6B   01   11   50   F0  01DE   206          insv    r0,#dtr_v_no_share,#1,(r11) ;set no share
                50   51   01   01   EF  01E3   207          extzv   #fab$v_shrget,#1,r1,r0      ;get shrget from shr
                6B   01   18   50   F0  01E8   208          insv    r0,#dtr_v_shared_read,#1,(r11) ;set shared read
                50   51   01   00   EF  01ED   209          extzv   #fab$v_shrput,#1,r1,r0      ;get shrput from shr
                6B   01   19   50   F0  01F2   210          insv    r0,#dtr_v_shared_write,#1,(r11) ;set shared write
                        5B   08   AB  9E  01F7   211          movab   8(r11),r11                  ;point to device bits
                        8B   40   A8  D0  01FB   212          movl    fab$l_dev(r8),(r11)+        ;set device bits
                        8B   44   A8  D0  01FF   213          movl    fab$l_sdc(r8),(r11)+        ;set spooling device bits
                6B   20514553 8F   D0  0203   214          movl    #^a/SEQ /,(r11)             ;assume seql
                      1D A8   10   91  020A   215          cmpb    #fab$c_rel,fab$b_org(r8)    ;is it seq?
                              0B   19  020E   216          blss    20$                         ;if lss, yes, cont
                              10   14  0210   217          bgtr    30$                         ;if gtr, no, its idx
                6B   204C4552 8F   D0  0212   218          movl    #^a/REL /,(r11)             ;set relative
                              07   11  0219   219          brb     30$                         ;cont
                6B   20584449 8F   D0  021B   220 20$:     movl    #^a/IDX /,(r11)             ;set indexed
                        5B   03   AB  9E  0222   221 30$:     movab   3(r11),r11                  ;update pointer
                             08   AA  9F  0226   222          pushab  xab$w_pro(r10)              ;set addr of protection bits
                             08   AC  9F  0229   223          pushab  8(ap)                       ;set addr of display
           00000000'GF       02   FB  022C   224          calls   #2,g^pli$$protvcha          ;process protection
                        5B   18   AB  9E  0233   225          movab   24(r11),r11                 ;point to file name
                    6B   00F9 C6   9B  0237   226          movzbw  <fcb_b_nam+nam$b_rsl>(r6),(r11) ;set size
                              05   12  023C   227          bneq    40$                         ;if neq, cont
                    6B   0101 C6   9B  023E   228          movzbw  <fcb_b_nam+nam$b_esl>(r6),(r11) ;set size
                6B   012E C6   8B   28  0243   229 40$:     movc3   (r11)+,fcb_b_esa(r6),(r11) ;set file name
                              04   04  0249   230          ret                                 ;return
                                        024A   231
                                        024A   232 ;++
                                        024A   233 ; pli$extend - perform explicit extend on a pl1 file
                                        024A   234 ; this routine extends the disk allocation of a pl1 file. if the file is
                                        024A   235 ; closed, it is opened with declare time attributes. the file is then
                                        024A   236 ; extended by the required number of blocks.
                                        024A   237 ;
                                        024A   238 ; inputs:
```

```
                                024A   239 ;       0(ap) - number of arguments (at least 2)
                                024A   240 ;       4(ap) - address of file control block
                                024A   241 ;       8(ap) - number of blocks to extend the file
                                024A   242 ; outputs:
                                024A   243 ;       none.
                                024A   244 ;--
                         000C   024A   245          .entry  pli$extend,^m<r2,r3>
                       0119  30  024C   246          bsbw    check_bis2                  ;check parms, open file
                 07 0C A2  01  E0  024F   247          bbs     #atr_v_opened,fcb_l_attr(r2),5$ ;if file opened, cont
                       011F  30  0254   248          bsbw    do_open                     ;open file
                   00000020  0257   249          .long   atr_m_output                ;for output
           00B6 C2  08 AC  D0  025B   250 5$:     movl    8(ap),<fcb_b_fab+fab$l_alq>(r2) ;set allocation size
                             0261   251          $extend fcb_b_fab(r2)               ;extend the file
                    04 50  E9  026C   252          blbc    r0,10$                      ;if lbc, fail
                 50    01  D0  026F   253          movl    #1,r0                       ;set success
                             04  0272   254          ret                                 ;return
           50    00000000'8F  D0  0273   255 10$:     movl    #pli$_rmsf,r0               ;set rms fab error
                       0118  31  027A   256          brw     fail                        ;and fail
                             027D   257
```

```
                                027D   259 ;++
                                027D   260 ; pli$flush - flush pl1 file
                                027D   261 ; this routine flushes a pl1 file. if the file is closed, no operation
                                027D   262 ; occurs. otherwise, all file buffers are flushed.
                                027D   263 ;
                                027D   264 ; inputs:
                                027D   265 ;       0(ap) - number of arguments (at least 1)
                                027D   266 ;       4(ap) - address of file control block
                                027D   267 ; outputs:
                                027D   268 ;       none
                                027D   269 ;--
                         0004   027D   270          .entry  pli$flush,^m<r2>
                     00E1   30  027F   271          bsbw    check_bis1                      ;check parms, open file
          0D 0C A2     01   E1  0282   272          bbc     #atr_v_opened,fcb_l_attr(r2),5$ ;if file not opened, cont
                                0287   273          $flush  fcb_b_rab(r2)                   ;flush the file
                   04 50   E9  0291   274          blbc    r0,10$                          ;if lbc, fail
                 50   01   D0  0294   275 5$:      movl    #1,r0                           ;set success
                           04  0297   276          ret                                     ;return
          50   00000000'8F   D0  0298   277 10$:     movl    #pli$_rmsr,r0                   ;set rms rab error
                     00F3   31  029F   278          brw     fail                            ;and fail
                                02A2   279
```

```
                                02A2    281 ;++
                                02A2    282 ; pli$next_volume - perform next_volume processing for a pl1 file
                                02A2    283 ; this routine performs next volume processing for a pl1 file. if the file
                                02A2    284 ; is closed, it is opened with declare time attributes. the rms nxtvol
                                02A2    285 ; service is then performed.
                                02A2    286 ;
                                02A2    287 ; inputs:
                                02A2    288 ;       0(ap) - number of arguments (at least 1)
                                02A2    289 ;       4(ap) - address of file control block
                                02A2    290 ; outputs:
                                02A2    291 ;       none
                                02A2    292 ;--
                        000C    02A2    293         .entry  pli$next_volume,^m<r2,r3>
                     00BC   30  02A4    294         bsbw    check_bis1                  ;check parms, open file
          07 0C A2     01   E0  02A7    295         bbs     #atr_v_opened,fcb_l_attr(r2),5$ ;if file opened, cont
                     00C7   30  02AC    296         bsbw    do_open                     ;open file
                    00000000    02AF    297         .long   0                           ;use dcl attr
                                02B3    298 5$:     $nxtvol fcb_b_rab(r2)               ;mount next volume
                  04 50    E9  02BD    299         blbc    r0,10$                      ;if lbc, fail
                50   01    D0  02C0    300         movl    #1,r0                       ;set success
                                04  02C3    301         ret                             ;return
          50    00000000'8F    D0  02C4    302 10$:    movl    #pli$_rmsr,r0               ;set rms rab error
                     00C7   31  02CB    303         brw     fail                        ;and fail
                                02CE    304
```

```
                                  02CE    306 ;++
                                  02CE    307 ; pli$rewind - rewind a pl1 file
                                  02CE    308 ; this routine rewinds a pl1 file. if the file is closed, it is opened with
                                  02CE    309 ; declare time attributes. the file is repositioned to its first record.
                                  02CE    310 ;
                                  02CE    311 ; inputs:
                                  02CE    312 ;       0(ap) - number of arguments (at least 1)
                                  02CE    313 ;       4(ap) - address of file control block
                                  02CE    314 ; outputs:
                                  02CE    315 ;       none
                                  02CE    316 ;--
                          000C    02CE    317         .entry  pli$rewind,^m<r2,r3>
                      0090   30   02D0    318         bsbw    check_bis1              ;check parms, open file
            07 0C A2    01   E0   02D3    319         bbs     #atr_v_opened,fcb_l_attr(r2),5$ ;if file opened, cont
                      009B   30   02D8    320         bsbw    do_open                 ;open file
                     00000000    02DB    321         .long   0                       ;use dcl attr
         0D 00E6 C2    02   E0   02DF    322 5$:     bbs     #dev$v_trm,<fcb_b_fab+fab$l_dev>(r2),20$ ;if term, cont
                                  02E5    323         $rewind fcb_b_rab(r2)           ;rewind the file
                   35 50   E9   02EF    324         blbc    r0,T0$                  ;if lbc, fail
               02040000 8F   C8   02F2    325 20$:    bisl    #<atr_m_currec!atr_m_virgin>,- ;set cur rec wrong
                     0C A2         02F8    326                 fcb_l_attr(r2)          ; and file in just opened state
         0C A2   0018000D 8F   CA   02FA    327         bicl    #<atr_m_delete!atr_m_recur! - ;clear delete, recursion flag,
                                  0302    328                 atr_m_comma_exp!atr_m_eof! - ;comma_expected, eof
                                  0302    329                 atr_m_write>,fcb_l_attr(r2) ;and write
                   20 A2   7C   0302    330         clrq    fcb_q_rfa(r2)           ;clear saved rfa
            1C 0C A2    0B   E1   0305    331         bbc     #atr_v_stream,fcb_l_attr(r2),30$ ;if not stream, cont
                   2E A2   B4   030A    332         clrw    fcb_w_column(r2)        ;clear column
            1C A2   14 A2   D0   030D    333         movl    fcb_l_buf(r2),fcb_l_buf_pt(r2) ;reset buffer pointer
            18 A2   14 A2   D0   0312    334         movl    fcb_l_buf(r2),fcb_l_buf_end(r2) ;reset buffer end
                50    01   9A   0317    335         movzbl  #1,r0                   ;get initial line, page and prn
                   30 A2    50   B0   031A    336         movw    r0,fcb_w_line(r2)       ;init line number
                   32 A2    50   B0   031E    337         movw    r0,fcb_w_page(r2)       ;init page number
                   34 A2    50   D0   0322    338         movl    r0,fcb_l_prn(r2)        ;init prn
                             04   0326    339 30$:    ret                             ;return
         50    00000000'8F   D0   0327    340 10$:    movl    #pli$_rmsr,r0           ;set rms rab error
                       0064   31   032E    341         brw     fail                    ;and fail
                              0331    342
```

```
                        0331   344 ;++
                        0331   345 ; pli$spaceblock - space blocks in a pl1 file
                        0331   346 ; this routine spaces forward or backward a specified number of blocks in
                        0331   347 ; a pl1 file. if the file is closed, it is opened with declare time attri-
                        0331   348 ; butes and the block_io environment attribute implied. the current block
                        0331   349 ; position in the file is moved the requested amount.
                        0331   350 ;
                        0331   351 ; inputs:
                        0331   352 ;       0(ap) - number of arguments (at least 2)
                        0331   353 ;       4(ap) - address of file control block
                        0331   354 ;       8(ap) - number of blocks to space
                        0331   355 ; outputs:
                        0331   356 ;       none
                        0331   357 ;--
                   000C 0331   358         .entry  pli$spaceblock,^m<r2,r3>
                0032 30 0333   359         bsbw    check_bis2                      ;check parms, open file
      07 0C A2    01 E0 0336   360         bbs     #atr_v_opened,fcb_l_attr(r2),5$ ;if file opened, cont
                0038 30 033B   361         bsbw    do_open                         ;open file
                00400000 033E  362         .long   atr_m_blockio                   ;set for blockio
   009A C2   08 AC D0 0342   363 5$:      movl    8(ap),<fcb_b_rab+rab$l_bkt>(r2) ;set number of blocks to space
                     0348   364         $space  fcb_b_rab(r2)                   ;space the file
             04 50 E9 0352   365         blbc    r0,10$                          ;if lbc, fail
          50 01 D0 0355   366         movl    #1,r0                           ;set success
                04 0358   367         ret                                     ;return
   50 00000000'8F D0 0359   368 10$:      movl    #pli$_rmsr,r0                   ;set rms rab error
                0032 31 0360   369         brw     fail                            ;and fail
                     0363   370
                     0363   371         .enabl  lsb
                     0363   372 check_bis1:
             01 6C D1 0363   373         cmpl    (ap),#1                         ;enought arguments?
                03 11 0366   374         brb     5$                              ;cont in common
                     0368   375 check_bis2:
             02 6C D1 0368   376         cmpl    (ap),#2                         ;enough arguments?
                04 18 036B   377 5$:      bgeq    10$                             ;if geq, yes, cont
                50 D4 036D   378         clrl    r0                              ;set not enough args
                24 11 036F   379         brb     fail                            ;and fail
          52 04 AC D0 0371   380 10$:     movl    4(ap),r2                        ;get addr of fcb
                05 0375   381         rsb                                     ;
                     0376   382         .dsabl  lsb
                     0376   383
                     0376   384 do_open:
             53 6E D0 0376   385         movl    (sp),r3                         ;get addr of parm
             6E 83 D0 0379   386         movl    (r3)+,(sp)                      ;set open attr
                52 DD 037C   387         pushl   r2                              ;set fcb addr
   00000000'GF 02 FB 037E   388         calls   #2,g^pli$open                   ;open file
      09 0C A2 01 E0 0385   389         bbs     #atr_v_opened,fcb_l_attr(r2),10$ ;if file still not open
   50 00000000'8F D0 038A   390         movl    #pli$_open,r0                   ;set open failure
                02 11 0391   391         brb     fail                            ;and fail
                63 17 0393   392 10$:     jmp     (r3)                            ;return
                     0395   393
                0B 12 0395   394 fail:   bneq    10$                             ;if not enough parms
   50 00000000'8F D0 0397   395         movl    #pli$_parm,r0                   ;set not enuough parms
                51 D4 039E   396         clrl    r1                              ;set no fcb addr
                03 11 03A0   397         brb     20$                             ;cont
             51 52 D0 03A2   398 10$:     movl    r2,r1                           ;copy fcb addr
                51 DD 03A5   399 20$:     pushl   r1                              ;set fcb addr
                50 DD 03A7   400         pushl   r0                              ;set error code
```

```
                00000000'8F   DD  03A9   401          pushl   #pli$_error             ;set error condition
            00000000'GF   03  FB  03AF   402          calls   #3,g^pli$io_error       ;signal the condition
                              04  03B6   403          ret                             ;return
                                  03B7   404
                                  03B7   405
                                  03B7   406          .end
```

```
$$.TMP1                         = 00000001
$$.TMP2                         = 000000A2
ATR_M_BLOCKIO                   = 00400000
ATR_M_COMMA_EXP                 = 00000004
ATR_M_CURREC                    = 00040000
ATR_M_DELETE                    = 00080000
ATR_M_EOF                       = 00000001
ATR_M_OUTPUT                    = 00000020
ATR_M_RECUR                     = 00000008
ATR_M_VIRGIN                    = 02000000
ATR_M_WRITE                     = 00100000
ATR_V_APP_COMMA                 = 00000015
ATR_V_INDEXED                   = 00000010
ATR_V_OPENED                    = 00000001
ATR_V_STREAM                    = 0000000B
ATR_V_UPDATE                    = 00000004
BLOCK_SIZE                        00000000
BUCKET_SIZE                       00000004
CHECK_BIS1                        00000363 R      02
CHECK_BIS2                        00000368 R      02
COLUMN_NUMBER                     0000006C
CREATION_DATE                     00000008
DAT_K_CHAR_VAR                  = 0000000B
DEV$V_TRM                       = 00000002
DEVICE                            0000007C
DO_OPEN                           00000376 R      02
DSPEND                            00000121
DTR                               00000074
DTR_M_FIXED_LENGTH_RECORDS      = 00000400
DTR_V_BLOCK_BOUNDARY_FORMAT     = 00000001
DTR_V_BLOCK_IO                  = 00000004
DTR_V_CARRIAGE_RETURN_FORMAT    = 00000006
DTR_V_FORTRAN_FORMAT            = 00000000
DTR_V_IGNORE_LINE_MARKS         = 0000000C
DTR_V_INDEXED                   = 00000029
DTR_V_INITIAL_FILL              = 00000010
DTR_V_NO_SHARE                  = 00000011
DTR_V_PRINTER_FORMAT            = 00000012
DTR_V_READ_AHEAD                = 00000016
DTR_V_SHARED_READ               = 00000018
DTR_V_SHARED_WRITE              = 00000019
DTR_V_UPDATE                    = 0000001D
DTR_V_WRITE_BEHIND              = 00000013
EXPANDED_TITLE                    0000009F
EXPIRATION_DATE                   00000010
EXTENSION_SIZE                    00000018
FAB$B_BKS                       = 0000003E
FAB$B_FAC                       = 00000016
FAB$B_FSZ                       = 0000003F
FAB$B_ORG                       = 0000001D
FAB$B_RAT                       = 0000001E
FAB$B_RFM                       = 0000001F
FAB$B_RTV                       = 0000001C
FAB$B_SHR                       = 00000017
FAB$C_FIX                       = 00000001
FAB$C_REL                       = 00000010
FAB$L_ALQ                       = 00000010
FAB$L_DEV                       = 00000040
FAB$L_FOP                       = 00000004
FAB$L_MRN                       = 00000038
FAB$L_SDC                       = 00000044
FAB$L_XAB                       = 00000024
FAB$M_CBT                       = 00200000
FAB$M_CTG                       = 00100000
FAB$M_DFW                       = 00000020
FAB$M_DLT                       = 00008000
FAB$M_POS                       = 00000100
FAB$M_RCK                       = 00800000
FAB$M_RWC                       = 00000800
FAB$M_RWO                       = 00000080
FAB$M_SCF                       = 00004000
FAB$M_SPL                       = 00002000
FAB$M_SUP                       = 00000004
FAB$M_TEF                       = 10000000
FAB$M_TMP                       = 00000008
FAB$M_WCK                       = 00000200
FAB$V_BIO                       = 00000005
FAB$V_BLK                       = 00000003
FAB$V_CR                        = 00000001
FAB$V_FTN                       = 00000000
FAB$V_NIL                       = 00000005
FAB$V_PRN                       = 00000002
FAB$V_SHRGET                    = 00000001
FAB$V_SHRPUT                    = 00000000
FAB$W_BLS                       = 0000003C
FAB$W_DEQ                       = 00000014
FAB$W_MRS                       = 00000036
FAIL                              00000395 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
FILE_ID                         0000001C
FILE_ORGANIZATION               00000084
FILE_SIZE                       00000034
FIXED_CONTROL_SIZE              00000038
GROUP_PROTECTION                00000087
INDEX_NUMBER                    0000003C
LINESIZE                        0000005C
LINE_NUMBER                     00000068
MAXIMUM_RECORD_NUMBER           00000040
MAXIMUM_RECORD_SIZE             00000044
MULTIBLOCK_COUNT                00000048
MULTIBUFFER_COUNT               0000004C
NAM$B_ESL                     = 0000000B
NAM$B_RSL                     = 00000003
NAM$T_DVI                     = 00000014
NUMBER_OF_KEYS                  00000070
OWNER_GROUP                     00000050
OWNER_MEMBER                    00000054
OWNER_PROTECTION                0000008D
PAGESIZE                        00000060
PAGE_NUMBER                     00000064
PLI$$BYTESIZE                   ********    X   02
PLI$$PROTVCHA                   ********    X   02
PLI$DISPLAY                     00000000 RG     02
PLI$EXTEND                      0000024A RG     02
PLI$FLUSH                       0000027D RG     02
PLI$IO_ERROR                    ********      X 02
PLI$NEXT_VOLUME                 000002A2 RG     02
PLI$OPEN                        ********    X   02
PLI$REWIND                      000002CE RG     02
PLI$SPACEBLOCK                  00000331 RG     02
PLI$_ERROR                      ********      X 02
PLI$_INVDATYP                   ********      X 02
PLI$_OPEN                       ********      X 02
PLI$_PARM                       ********      X 02
PLI$_RMSF                       ********      X 02
PLI$_RMSR                       ********      X 02
RAB$B_KRF                     = 00000035
RAB$B_MBC                     = 00000037
RAB$B_MBF                     = 00000036
RAB$L_BKT                     = 00000038
RAB$L_ROP                     = 00000004
RAB$V_LOA                     = 0000000D
RAB$V_RAH                     = 00000009
RAB$V_WBH                     = 0000000A
RETRIEVAL_POINTERS              00000058
SIZ...                        = 00000001
SPOOL_DEVICE                    00000080
SYS$DISPLAY                     ********  GX    02
SYS$EXTEND                      ********  GX    02
SYS$FLUSH                       ********  GX    02
SYS$NXTVOL                      ********  GX    02
SYS$REWIND                      ********  GX    02
SYS$SPACE                       ********  GX    02
SYSTEM_PROTECTION               00000093
WORLD_PROTECTION                00000099
XAB$B_BLN                     = 00000001
XAB$B_COD                     = 00000000
XAB$C_DAT                     = 00000012
XAB$C_FHC                     = 0000001D
XAB$C_PRO                     = 00000013
XAB$K_DATLEN                  = 0000002C
XAB$K_FHCLEN                  = 0000002C
XAB$K_PROLEN                  = 00000058
XAB$L_HBK                     = 0000000C
XAB$L_NXT                     = 00000004
XAB$Q_CDT                     = 00000014
XAB$Q_EDT                     = 0000001C
XAB$W_GRP                     = 0000000E
XAB$W_MBM                     = 0000000C
XAB$W_PRO                     = 00000008
```

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	000003B7 (951.)	02 (2.)	PIC USR CON REL LCL SHR EXE RD NOWRT NOVEC LONG

Performance indicators

Phase	Page faults	CPU Time	Elapsed Time
Initialization	12	00:00:00.05	00:00:00.34
Command processing	98	00:00:00.58	00:00:01.95
Pass 1	260	00:00:11.28	00:00:23.48
Symbol table sort	0	00:00:01.30	00:00:02.47
Pass 2	79	00:00:02.05	00:00:04.87
Symbol table output	22	00:00:00.17	00:00:00.39
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	473	00:00:15.47	00:00:33.54

The working set limit was 1200 pages.
60982 bytes (120 pages) of virtual memory were used to buffer the intermediate code.
There were 60 pages of symbol table space allocated to hold 971 non-local and 25 local symbols.
406 source lines were read in Pass 1, producing 29 object records in Pass 2.
29 pages of virtual memory were used to define 26 macros.

Macro library statistics

Macro library name	Macros defined
_255DUA28:[PLIRTL.OBJ]PLIRTMAC.MLB;1	4
_255DUA28:[SYSLIB]STARLET.MLB;2	19
TOTALS (all libraries)	23

987 GETS were required to define 23 macros.

There were no errors, warnings or information messages.

MACRO/ENABLE=SUPPRESSION/DISABLE=TRACEBACK/LIS=LIS$:PLIRMSBIS/OBJ=OBJ$:PLIRMSBIS MSRC$:PLIRMSBIS/UPDATE=(ENH$:PLIRMSBIS)+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

BANNER
LIS

PLIWRITE
LIS

SMBREQ
REQ

PLIVECTOR
LIS

SMBSRVSHR
MAP

FORMAT
LIS

PLIRODATA
LIS

SMBDEF
SDL

PRTSMB

DISPATCH
L13

PLISTRING
LIS

PRTSMB
MAP