```
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
```

```
PPPPPPPP    LL          IIIIII    EEEEEEEEEE  NN      NN  VV      VV    IIIIII    RRRRRRRR
PPPPPPPP    LL          IIIIII    EEEEEEEEEE  NN      NN  VV      VV    IIIIII    RRRRRRRR
PP      PP  LL            II      EE          NN      NN  VV      VV      II      RR      RR
PP      PP  LL            II      EE          NN      NN  VV      VV      II      RR      RR
PP      PP  LL            II      EE          NNNN    NN  VV      VV      II      RR      RR
PP      PP  LL            II      EE          NNNN    NN  VV      VV      II      RR      RR
PPPPPPPP    LL            II      EEEEEEEE    NN  NN  NN  VV      VV      II      RRRRRRRR
PPPPPPPP    LL            II      EEEEEEEE    NN  NN  NN  VV      VV      II      RRRRRRRR
PP          LL            II      EE          NN    NNNN  VV      VV      II      RR  RR
PP          LL            II      EE          NN    NNNN  VV      VV      II      RR  RR
PP          LL            II      EE          NN      NN    VV  VV        II      RR    RR        ....
PP          LL            II      EE          NN      NN    VV  VV        II      RR    RR        ....
PP          LLLLLLLLLL  IIIIII    EEEEEEEEEE  NN      NN      VV        IIIIII    RR      RR      ....
PP          LLLLLLLLLL  IIIIII    EEEEEEEEEE  NN      NN      VV        IIIIII    RR      RR      ....


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
```

```
/*                                                          EDIT: HE2004
*******************************************************************************
**                                                                           *
**  COPYRIGHT (c) 1978, 1980, 1982, 1984 BY                                  *
**  DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.                   *
**  ALL RIGHTS RESERVED.                                                     *
**                                                                           *
**  THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED    *
**  ONLY IN  ACCORDANCE WITH  THE  TERMS  OF  SUCH  LICENSE  AND WITH THE    *
**  INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR  ANY  OTHER    *
**  COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY    *
**  OTHER PERSON.  NO TITLE TO AND OWNERSHIP OF  THE  SOFTWARE IS  HEREBY    *
**  TRANSFERRED.                                                             *
**                                                                           *
**  THE INFORMATION IN THIS SOFTWARE IS  SUBJECT TO CHANGE WITHOUT NOTICE    *
**  AND  SHOULD  NOT  BE  CONSTRUED AS  A COMMITMENT BY DIGITAL EQUIPMENT    *
**  CORPORATION.                                                             *
**                                                                           *
**  DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE  OR  RELIABILITY OF ITS    *
**  SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.                  *
**                                                                           *
**                                                                           *
*******************************************************************************

facility:       VAX-11 PL/I Runtime Library.

abstract:       This routine is called to process the environment attributes
                for the PL/I open service.

author:         C. Spitz

date:           23-Jan-1980

Modifications:
    V1.4-02:    Bill Matthews   28-Sep-1981

                Fix to not maximize versions ever when an explicit version
                number is specified.

    V1.4-03:    Bill Matthews   08-Oct-1981

                Fix coding of protection utility to not rely on short circuit
                boolean optimization for correct execution of the program.

    V2.0-04:    Hisham Elbasha  11-NOV-1982

                make the upi bit independent of the bio bit for shared_read
                and shared_write.
*/
/*
Local Commentary:
        The environment options for a file may be specified on the DECLARE
        statement for the file, on the OPEN statement, or on the CLOSE
        statement.  The environment options are represented as a list of
        elements, where each element is represented by its type code, and
```

```
                its value. The type code is one byte long; valid environments have
                values of 1 through num_envir_opts. The value of 0 is used to des-
                ignate the end of the environment list. Each environment option has
                a parameter, whose interpretation is dependant upon the option. The
                parameters data types are:
                        immediate bit - represented as 1 byte, low bit = value
                        immediate value - represented as 1 longword
                        immediate character - represented as n bytes. the first
                                2 bytes are the total length of the character
                                string, the second 2 bytes are the current length
                                of the character sting, and the remaining n-4 bytes
                                are storage for the total length of the string. Note
                                that both lengths do not include the length fields.
                        address - represented as a 4 byte absolute address.
                        quad value - represented as a 4 byte absolute address. */

        pli$$envir: procedure(fcbpt,openv,open_blk) options(ident('1-004'))
                        returns(fixed bin(31));

    1
    1   /* parameter declarations */
    1   dcl     fcbpt                   pointer, /* pointer to file control block */
    1           openv                   pointer, /* pointer to open environment   */
    1           open_blk                pointer; /* pointer to open block         */
    1
    1   /* the following is a template for the macro open block */
    1   dcl     1 opn                   based(open_blk),
    1                   2 status(0:31)  bit,
    1                   2 create_date(0:1) fixed bin(31),
    1                   2 expire_date(0:1) fixed bin(31),
    1                   2 file_id_to_pt pointer,
    1                   2 fixed_control_to_pt pointer,
    1                   2 prot(0:15)    bit,
    1                   2 own_group     fixed bin(15),
    1                   2 own_mem       fixed bin(15);
    1           /* bit offsets for status */
    1           %replace create_dat     by 0;
    1           %replace expire_dat     by 1;
    1           %replace fileid_to      by 2;
    1           %replace fixedctl_to    by 3;
    1           %replace protect        by 4;
    1           %replace uic            by 5;
    1           %replace close          by 6;
    1           /* bit offsets for protection */
    1           %replace no_read by 0;
    1           %replace no_write by 1;
    1           %replace no_execute by 2;
    1           %replace no_delete by 3;
    1           %replace system_prot by 0;
    1           %replace owner_prot by 4;
    1           %replace group_prot by 8;
    1           %replace world_prot by 12;
    1
    1   /* general constants */
    1   %replace true                   by '1'b;
    1   %replace false                  by '0'b;
    1
    1   /* global declarations */
```

```
  113    1      %include envcodes; /* define environment codes and types  */
  171    1      %include filedef; /* define file control block, fab, rab, nam*/
  408    1
```

```
409 ! 1     /* local data - static */
410 ! 1     /* the following table contains the parameter type for each environment option*/
411   1     %replace bittyp                 by 0;
412   1     %replace longtyp                by 1;
413   1     %replace quadtyp                by 2;
414   1     %replace stringtyp              by 3;
415   1     %replace addrtyp                by 4;
416   1     dcl     env_type(num_envir_opts) fixed bin(7) static readonly
417   1                     init(   bittyp,         /* append */
418   1                             bittyp,         /* batch */
419   1                             bittyp,         /* block_boundry */
420   1                             bittyp,         /* block_io */
421   1                             longtyp,        /* block_size */
422   1                             longtyp,        /* bucket_size */
423   1                             bittyp,         /* carriage */
424   1                             bittyp,         /* contiguous */
425   1                             bittyp,         /* contiguous_best_try */
426   1                             quadtyp,        /* creation_date */
427   1                             bittyp,         /* current_position */
428   1                             stringtyp,      /* default_file_name */
429   1                             bittyp,         /* defered_write */
430   1                             bittyp,         /* delete */
431   1                             quadtyp,        /* expiration_date */
432   1                             longtyp,        /* extension_size */
433   1                             addrtyp,        /* file_id */
434   1                             addrtyp,        /* file_id_to */
435   1                             longtyp,        /* file_size */
436   1                             longtyp,        /* fixed_control_size */
437   1                             addrtyp,        /* fixed_control_size_to */
438   1                             bittyp,         /* fixed_length_records */
439   1                             stringtyp,      /* group_protection */
440   1                             bittyp,         /* ignore_line_marks */
441   1                             bittyp,         /* indexed */
442   1                             bittyp,         /* indexed_fill */
443   1                             longtyp,        /* index_number */
444   1                             longtyp,        /* max_record_number */
445   1                             longtyp,        /* max_record_size */
446   1                             longtyp,        /* multiblock_count */
447   1                             longtyp,        /* multibuffer_count */
448   1                             bittyp,         /* no_share */
449   1                             longtyp,        /* owner_group */
450   1                             longtyp,        /* owner_member */
451   1                             stringtyp,      /* owner_protection */
452   1                             bittyp,         /* printer */
453   1                             bittyp,         /* read_ahead */
454   1                             bittyp,         /* read_check */
455   1                             bittyp,         /* record_id_access */
456   1                             longtyp,        /* retreityp_pointers */
457   1                             bittyp,         /* rewind_close */
458   1                             bittyp,         /* rewind_open */
459   1                             bittyp,         /* scalarvarying */
460   1                             bittyp,         /* shared_read */
461   1                             bittyp,         /* shared_write */
462   1                             bittyp,         /* spool */
463   1                             bittyp,         /* supersede */
464   1                             stringtyp,      /* system_protection */
```

```
  465    1                              bittyp,          /* temporary */
  466    1                              bittyp,          /* truncate */
  467    1                              stringtyp,       /* world_protection */
  468    1                              bittyp,          /* write_behind */
  469    1                              bittyp);         /* write_check */
  470    1
  471    1       dcl     1 end_opt                       static readonly, /* end of environment list */
  472    1                       2 env_number            fixed bin(7) init(unused_envir_opt);
  473    1
  474    1       dcl     default_name                    char(4) static readonly init('.DAT');
  475    1
```

```
476 ! 1     /* local data - automatic */
477   1     dcl     fcb                       pointer, /* local pointer to fcb (unaliased) */
478   1             declared_environment      pointer,
479   1             current_env_number        fixed bin(7),
480   1             next_specified_env_number fixed bin(7),
481   1             longtemp                  fixed bin(31),
482   1             point                     pointer, /* utility pointer */
483   1             error_code                fixed bin(31),
484   1             carriage_specified_false bit aligned, /* true if carriage was specified
485 ! 1                                                     as '0'b */
486   1             specified                 bit aligned; /* true if current_env_number was
487 ! 1                                                     specified in an environment list */
488   1
489 ! 1     /* the following are used to compare the declared and open environments, to
490 ! 1        ensure that they are the same. THEY ARE NOT AVAILABLE FOR USE AS TEMPS. */
491   1     dcl     bitval(0:1)               bit aligned,
492   1             addrval(0:1)              pointer,
493   1             longval(0:1)              fixed bin(31),
494   1             quadval(0:1,0:1)          fixed bin(31);
495 ! 1     /*
496 ! 1
```

```
497 ! 1     /* based declarations */
498 ! 1     /* the following declarations are templates for the various types of environment
499 ! 1           options. there is one template for each parameter type. */
500   1     dcl   1 optbit                  based,
501   1             2 env_number    fixed bin(7),
502   1             2 bitf          bit,
503   1             2 bitext(7)     bit,
504   1             2 bitnext       fixed bin(7);
505   1     dcl   1 optlong                 based,
506   1             2 env_number    fixed bin(7),
507   1             2 long          fixed bin(31),
508   1             2 longnext      fixed bin(7);
509   1     dcl   1 optaddr                 based,
510   1             2 env_number    fixed bin(7),
511   1             2 address       pointer,
512   1             2 addrnext      fixed bin(7);
513   1     dcl   1 optstring               based,
514   1             2 env_number    fixed bin(7),
515   1             2 maxsize       fixed bin(15),
516   1             2 string        char(128) var;
517   1     dcl   1 optstringnext           based,
518   1             2 env_number    fixed bin(7),
519   1             2 maxsize       fixed bin(15),
520   1             2 cursize       fixed bin(15),
521   1             2 stringnext(0:128) fixed bin(7);
522   1
523 ! 1     /* the following are templates for moving values around */
524   1     dcl   value                     fixed bin(31) based;
525   1     dcl   qvalue(0:1)               fixed bin(31) based;
526   1     dcl   byte                      fixed bin(7) based(addr(longval));
527   1     dcl   word                      fixed bin(15) based(addr(longval));
528   1     dcl   fileid                    char(22) based(addrval(0));
529   1     dcl   bytetemp                  fixed bin(7) based(addr(longtemp));
530   1     dcl   wordtemp                  fixed bin(15) based(addr(longtemp));
531   1     dcl   buflen                    fixed bin(15) based(
532   1                                       addr(fcb->file_constant.buffer_end));
533   1     dcl   stringtemp                char(128) var based;
534   1     dcl   1 s                       based,
535   1             2 stringlen     fixed bin(15),
536   1             2 stringval     char(128);
537   1
```

```
538 :  1      /* declarations of error messages and error routines */
539    1      dcl     pli$io_error            entry(fixed bin(31) value,
540    1                                      fixed bin(31) value,pointer value);
541    1      dcl     pli$_undfile            globalref fixed bin(31) value;
542    1      dcl     pli$_envparm            globalref fixed bin(31) value;
543    1      dcl     pli$_invdfnam           globalref fixed bin(31) value;
544    1      dcl     pli$_conappsup          globalref fixed bin(31) value;
545    1      dcl     pli$_conblokio          globalref fixed bin(31) value;
546    1      dcl     pli$_invrtvptr          globalref fixed bin(31) value;
547    1      dcl     pli$_noshare            globalref fixed bin(31) value;
548    1      dcl     pli$_invprot            globalref fixed bin(31) value;
549    1      dcl     pli$_invmltblk          globalref fixed bin(31) value;
550    1      dcl     pli$_invmltbuf          globalref fixed bin(31) value;
551    1      dcl     pli$_confixlen          globalref fixed bin(31) value;
552    1      dcl     pli$_invindnum          globalref fixed bin(31) value;
553    1      dcl     pli$_invblksiz          globalref fixed bin(31) value;
554    1      dcl     pli$_invbktsiz          globalref fixed bin(31) value;
555    1      dcl     pli$_invextsiz          globalref fixed bin(31) value;
556    1      dcl     pli$_invfxcsiz          globalref fixed bin(31) value;
557    1      dcl     pli$_conenvopt          globalref fixed bin(31) value;
558    1      dcl     pli$_conprintcr         globalref fixed bin(31) value;
559    1      dcl     pli$_invowngrp          globalref fixed bin(31) value;
560    1      dcl     pli$_invownmem          globalref fixed bin(31) value;
561    1      dcl     pli$_conprtfrm          globalref fixed bin(31) value;
562    1      dcl     pli$_creindex           globalref fixed bin(31) value;
563    1      dcl     pli$_invmaxrec          globalref fixed bin(31) value;
564    1
```

```
565 ¦  1        /* initialization */
566 ¦  1        /* define general error condition handler */
567    1        on anycondition begin;
568    2                error_code = pli$_envparm;
569    2                goto opt_error;
570    2                end;
571    1
572    1        fcb = fcbpt; /* copy fcb pointer to local storage */
573    1        declared_environment = addr(fcb -> fcb_end); /* point to declared environment */
574    1        if openv = null()
575    1                then openv = addr(end_opt);
576    1        if fcb -> fcb_end = 0 ¦ opn.status(close)
577    1                then declared_environment = addr(end_opt);
578    1        next_specified_env_number = 0;
579    1
```

```
580 ! 1     /* main loop */
581   1     do current_env_number = 0 to num_envir_opts;
582   2             specified = (next_specified_env_number = current_env_number);
583   2             if opn.status(close)
584   2                     then do;
585   2                             if current_env_number = batch !
586   2                                     current_env_number = delete !
587   2                                     current_env_number = rewind_close !
588   2                                     current_env_number = spool !
589   2                                     current_env_number = truncate
590   2                                 then goto opt(current_env_number);
591   2                             end;
592   2                     else goto opt(current_env_number);
593   2             goto next_opt;
594   2
595 ! 2     /* error routine */
596   2
597   2     opt_error:
598   2             revert anycondition;
599   2             call pli$io_error(pli$_undfile,error_code,fcb);
600   2             return(pli$_undfile);
601   2
```

```
602     2       opt(0):
603     2               goto next_opt;
604     2       opt(append):
605     2               if specified & bitval(0)
606     2                       then do;
607     3                               fcb -> attr(atr_v_app) = true;
608     3                               fcb -> fab$l_fop(fab$v_mxv) = false;
609     3                               fcb -> fab$l_fop(fab$v_cif) = true;
610     3                               fcb -> fab$l_fop(fab$v_sup) = false;
611     3                               fcb -> fab$l_fop(fab$v_nef) = false;
612     3                               fcb -> rab$l_rop(rab$v_eof) = true;
613     3                               end;
614     2                       else fcb -> attr(atr_v_app) = false;
615     2               goto next_opt;
616     2
617     2
618     2       opt(batch):
619     2               fcb -> fab$l_fop(fab$v_scf) = specified & bitval(0);
620     2               goto next_opt;
621     2
622     2
623     2       opt(block_boundry):
624     2               fcb -> fab$b_rat(fab$v_blk) = specified & bitval(0);
625     2               goto next_opt;
626     2
627     2
628     2       opt(block_io):
629     2               if specified & bitval(0) ¦ fcb -> attr(atr_v_blockio)
630     2                       then do;
631     3                               if fcb -> fab$b_rat(fab$ lk)
632     3                                       ¦ fcb -> tr(atr_v_stream)
633     3                                       then do;
634     4                                               error_code = pli$_conblokio;
635     4                                               goto opt_error;
636     4                                               end;
637     3                               fcb -> fab$b_fac(fab$v_bio) = true;
638     3                               fcb -> fab$b_rfm = fab$c_udf;
639     3                               end;
640     2                       else    fcb -> fab$b_fac(fab$v_bio) = false;
641     2               fcb -> fab$b_shr(fab$v_upi) = false;
642     2               goto next_opt;
643     2
644     2
645     2       opt(block_size):
646     2               if specified
647     2                       then do;
648     3                               if longval(0) < 0 ¦ longval(0) > 65535
649     3                                       then do;
650     4                                               error_code = pli$_invblksiz;
651     4                                               goto opt_error;
652     4                                               end;
653     3                               fcb -> fab$w_bls = word;
654     3                               end;
655     2                       else fcb -> fab$w_bls = 0;
656     2               goto next_opt;
657     2
```

```
658     2
659     2       opt(bucket_size):
660     2               if specified
661     2                       then do;
662     3                               if longval(0) < 0 ! longval(0) > 32
663     3                                       then do;
664     4                                               error_code = pli$_invbktsiz;
665     4                                               goto opt_error;
666     4                                               end;
667     3                               fcb -> fab$b_bks = byte;
668     3                               end;
669     2                       else fcb -> fab$b_bks = 0;
670     2               goto next_opt;
671     2
672     2
673     2       opt(carriage):
674     2               if specified & bitval(0)
675     2                       then do;
676     3                               if fcb -> attr(atr_v_print)
677     3                                       then do;
678     4                                               error_code = pli$_conprintcr;
679     4                                               goto opt_error;
680     4                                               end;
681     3                               if fcb -> fab$b_fac(fab$v_bio)
682     3                                       then do;
683     4                                               error_code = pli$_conblokio;
684     4                                               goto opt_error;
685     4                                               end;
686     3                               fcb -> fab$b_rat(fab$v_cr) = true;
687     3                               end;
688     2                       else do;
689     3                               fcb -> fab$b_rat(fab$v_cr) = false;
690     3                               carriage_specified_false = specified;
691     3                               end;
692     2               fcb -> fab$b_rat(fab$v_ftn) = false;
693     2               goto next_opt;
694     2
695     2
696     2       opt(contiguous):
697     2               fcb -> fab$l_fop(fab$v_ctg) = specified & bitval(0);
698     2               goto next_opt;
699     2
700     2
701     2       opt(contiguous_best_try):
702     2               fcb -> fab$l_fop(fab$v_cbt) = specified & bitval(0);
703     2               goto next_opt;
704     2
705     2
706     2       opt(creation_date):
707     2               if specified
708     2                       then do;
709     3                               create_date(0) = quadval(0,0);
710     3                               create_date(1) = quadval(0,1);
711     3                               opn.status(create_dat) = true;
712     3                               end;
713     2               goto next_opt;
714     2
```

```
715     2
716     2       opt(current_position):
717     2               fcb -> fab$l_fop(fab$v_pos) = specified & bitval(0);
718     2               goto next_opt;
719     2
720     2
721     2       opt(default_file_name):
722     2               if specified
723     2                       then do;
724     3                               if addrval(0) -> stringlen > 128
725     4                                       then do;
726     5                                               error_code = pli$_invdfnam;
727     5                                               goto opt_error;
728     5                                               end;
729     4                               fcb -> fab$l_dna = addr(addrval(0) -> stringval);
730     4                               longtemp = addrval(0) -> stringlen;
731     4                               fcb -> fab$b_dns = bytetemp;
732     4                               end;
733     3                       else do;
734     4                               fcb -> fab$l_dna = addr(default_name);
735     4                               fcb -> fab$b_dns = length(default_name);
736     3               end;
737     2               goto next_opt;
738     2
739     2
740     2       opt(defered_write):
741     2               fcb -> fab$l_fop(fab$v_dfw) = specified & bitval(0);
742     2               goto next_opt;
743     2
744     2
745     2       opt(delete):
746     2               fcb -> fab$l_fop(fab$v_dlt) = specified & bitval(0);
747     2               goto next_opt;
748     2
749     2
750     2       opt(expiration_date):
751     2               if specified
752     2                       then do;
753     3                               expire_date(0) = quadval(0,0);
754     3                               expire_date(1) = quadval(0,1);
755     3                               opn.status(expire_dat) = true;
756     3                               end;
757     2               goto next_opt;
758     2
759     2
760     2       opt(extension_size):
761     2               if specified
762     2                       then do;
763     3                               if longval(0) < 0 ! longval(0) > 65535
764     4                                       then do;
765     5                                               error_code = pli$_invextsiz;
766     5                                               goto opt_error;
767     5                                               end;
768     4                               fcb -> fab$w_deq = word;
769     4                               end;
770     3                       else fcb -> fab$w_deq = 0;
771     2               goto next_opt;
```

```
772     2
773     2
774     2       opt(file_id):
775     2               if specified
776     2                       then do;
777     3                               fcb -> nam$t_dvi = fileid;
778     3                               fcb -> nam$w_did = 0;
779     3                               fcb -> nam$w_did_seq = 0;
780     3                               fcb -> nam$w_did_rvn = 0;
781     3                               fcb -> fab$l_fop(fab$v_nam) = true;
782     3                               end;
783     2                       else fcb -> fab$l_fop(fab$v_nam) = false;
784     2               goto next_opt;
785     2
786     2
787     2       opt(file_id_to):
788     2               if specified
789     2                       then do;
790     3                               file_id_to_pt = addrval(0);
791     3                               opn.status(fileid_to) = true;
792     3                               end;
793     2               goto next_opt;
794     2
795     2
796     2       opt(file_size):
797     2               if specified
798     2                       then fcb -> fab$l_alq = longval(0);
799     2                       else fcb -> fab$l_alq = 0;
800     2               goto next_opt;
801     2
802     2
803     2       opt(fixed_control_size):
804     2               if specified
805     2                       then do;
806     3                               if fcb -> attr(atr_v_stream) !
807     4                                       fcb -> attr(atr_v_update) !
808     4                                       fcb -> fab$b_fac(fab$v_bio) !
809     4                                       longval(0) < 0 ! longval(0) > 255
810     4                                       then do;
811     4                                               error_code = pli$_invfxcsiz;
812     4                                               goto opt_error;
813     4                                               end;
814     3                               fcb -> fab$b_fsz = byte;
815     3                               fcb -> fab$b_rfm = fab$c_vfc;
816     3                               end;
817     2                       else do;
818     3                               if fcb -> attr(atr_v_print)
819     4                                       then do;
820     4                                               fcb -> fab$b_fsz = 2;
821     4                                               fcb -> fab$b_rfm = fab$c_vfc;
822     4                                               fcb -> fab$b_rat(fab$v_prn) = true;
823     4                                               end;
824     3                                       else fcb -> fab$b_fsz = 0;
825     3                               end;
826     2               goto next_opt;
827     2
828     2
```

```
829  2      opt(fixed_control_size_to):
830  2              if specified
831  2                      then do;
832  3                              fixed_control_to_pt = addrval(0);
833  3                              opn.status(fixedctl_to) = true:
834  3                              end;
835  2              goto next_opt;
836  2
837  2
838  2      opt(fixed_length_records):
839  2              if specified & bitval(0)
840  2                      then do;
841  3                              if (fcb -> attr(atr_v_stream) &
842  3                                      fcb -> attr(atr_v_output)) |
843  3                                      (fcb -> fab$b_rfm = fab$c_vfc) |
844  3                                      (fcb -> fab$b_fac(fab$v_bio))
845  3                                      then do;
846  4                                              error_code = pli$_confixlen;
847  4                                              goto opt_error;
848  4                                              end;
849  3                              fcb -> fab$b_rfm = fab$c_fix;
850  3                              end;
851  2              goto next_opt;
852  2
853  2
854  2      opt(group_protection):
855  2              longtemp = group_prot;
856  2              goto protection;
857  2
858  2
859  2      opt(ignore_line_marks):
860  2              fcb -> attr(atr_v_app_comma) = ^(specified & bitval(0));
861  2              goto next_opt;
862  2
863  2
864  2      opt(indexed):
865  2              if specified & bitval(0)
866  2                      then do;
867  3                              if fcb -> attr(atr_v_output) & ^fcb -> attr(atr_v_app)
868  3                                      then do;
869  4                                              error_code = pli$_creindex;
870  4                                              goto opt_error;
871  4                                              end;
872  3                              fcb -> attr(atr_v_indexed) = true;
873  3                              fcb -> fab$b_org = fab$c_idx;
874  3                              end;
875  2                      else do;
876  3                              if fcb -> attr(atr_v_keyed) &
877  3                                      ^fcb -> fab$b_fac(fab$v_bio)
878  3                                      then fcb -> fab$b_org = fab$c_rel;
879  3                                      else fcb -> fab$b_org = fab$c_seq;
880  3                              end;
881  2              goto next_opt;
882  2
883  2
884  2      opt(indexed_fill):
885  2              fcb -> rab$l_rop(rab$v_loa) = specified & bitval(0);
```

```
886     2               goto next_opt;
887     2
888     2
889     2       opt(index_number):
890     2               if specified
891     2                       then do;
892     3                               if longval(0) > 255
893     3                                       then do;
894     4                                               error_code = pli$_invindnum;
895     4                                               goto opt_error;
896     4                                               end;
897     3                               fcb -> rab$b_krf = byte;
898     3                               end;
899     2                       else fcb -> rab$b_krf = 0;
900     2               goto next_opt;
901     2
902     2
903     2       opt(max_record_number):
904     2               if specified
905     2                       then fcb -> fab$l_mrn = longval(0);
906     2                       else fcb -> fab$l_mrn = 0;
907     2               goto next_opt;
908     2
909     2
910     2       opt(max_record_size):
911     2               wordtemp = 0;
912     2               bytetemp = fcb -> fab$b_fsz;
913     2               if fcb -> fab$b_org = fab$c_rel
914     2                       then buflen = 480 - wordtemp;
915     2                       else do;
916     3                               if fcb -> fab$b_rfm = fab$c_fix
917     3                                       then buflen = 512;
918     3                                       else buflen = 510 - wordtemp;
919     3                               end;
920     2                       if specified
921     2                               then do;
922     3                                       if longval(0) < 0 ¦ longval(0) > 32767
923     3                                               ¦ (fcb -> fab$b_org = fab$c_rel &
924     3                                                       longval(0) > 16383)
925     3                                               then do;
926     4                                                       error_code = pli$_invmaxrec;
927     4                                                       goto opt_error;
928     4                                                       end;
929     3                                       fcb -> fab$w_mrs = word;
930     3                                       end;
931     2                               else fcb -> fab$w_mrs = buflen;
932     2                       buflen = max(buflen,fcb -> fab$w_mrs);
933     2                       goto next_opt;
934     2
935     2
936     2       opt(multiblock_count):
937     2               if specified
938     2                       then do;
939     3                               if longval(0) < 0 ¦ longval(0) > 127
940     3                                       then do;
941     4                                               error_code = pli$_invmltblk;
942     4                                               goto opt_error;
```

```
                                        end;
                                fcb -> rab$b_mbc = byte;
                                end;
                        else fcb -> rab$b_mbc = 0;
                goto next_opt;


        opt(multibuffer_count):
                if specified
                        then do;
                                if longval(0) < 0 ! longval(0) > 127
                                        then do;
                                                error_code = pli$_invmltbuf;
                                                goto opt_error;
                                                end;
                                fcb -> rab$b_mbf = byte;
                                end;
                        else fcb -> rab$b_mbf = 0;
                goto next_opt;


        opt(no_share):
                fcb -> fab$b_shr(fab$v_nil) = specified & bitval(0);
                goto next_opt;


        opt(owner_group):
                if specified
                        then do;
                                if longval(0) < 0 ! longval(0) > 255
                                        then do;
                                                error_code = pli$_invowngrp;
                                                goto opt_error;
                                                end;
                                own_group = word;
                                opn.status(uic) = true;
                                end;
                        goto next_opt;


        opt(owner_member):
                if specified
                        then do;
                                if longval(0) < 0 ! longval(0) > 255
                                        then do;
                                                error_code = pli$_invownmem;
                                                goto opt_error;
                                                end;
                                own_mem = word;
                                opn.status(uic) = true;
                                end;
                        goto next_opt;


        opt(owner_protection):
                longtemp = owner_prot;
                goto protection;
```

```
1000     2
1001     2
1002     2          opt(printer):
1003     2                  if specified & bitval(0)
1004     2                          then do;
1005     2                                  if fcb -> attr(atr_v_stream) !
1006     2                                          fcb -> fab$b_rfm = fab$c_fix !
1007     2                                          fcb -> fab$b_rat(fab$v_cr) !
1008     2                                          fcb -> fab$b_fac(fab$v_bio)
1009     2                                      then do;
1010     2                                          error_code = pli$_conprtfrm;
1011     2                                          goto opt_error;
1012     2                                          end;
1013     2                                  fcb -> fab$b_rat(fab$v_prn) = true;
1014     2                                  fcb -> fab$b_rfm = fab$c_vfc;
1015     2                                  end;
1016     2                          else fcb -> fab$b_rat(fab$v_cr) = ^(fcb -> attr(atr_v_print) !
1017     2                                                  carriage_specified_false);
1018     2                  goto next_opt;
1019     2
1020     2
1021     2          opt(read_ahead):
1022     2                  fcb -> rab$l_rop(rab$v_rah) = true;
1023     2                  if specified
1024     2                          then fcb -> rab$l_rop(rab$v_rah) = bitval(0);
1025     2                  goto next_opt;
1026     2
1027     2
1028     2          opt(read_check):
1029     2                  fcb -> fab$l_fop(fab$v_rck) = specified & bitval(0);
1030     2                  goto next_opt;
1031     2
1032     2
1033     2          opt(record_id_access):
1034     2                  if specified & bitval(0) & fcb -> fab$b_fac(fab$v_bio)
1035     2                          then do;
1036     2                                  error_code = pli$_conblokio;
1037     2                                  goto opt_error;
1038     2                                  end;
1039     2                  fcb -> attr(atr_v_recidacc) = specified & bitval(0);
1040     2                  goto next_opt;
1041     2
1042     2
1043     2          opt(retreival_pointers):
1044     2                  if specified
1045     2                          then do;
1046     2                                  if longval(0) > 127 ! longval(0) < -1
1047     2                                      then do;
1048     2                                          error_code = pli$_invrtvptr;
1049     2                                          goto opt_error;
1050     2                                          end;
1051     2                                  if longval(0) = -1
1052     2                                      then longval(0) = 255;
1053     2                                  fcb -> fab$b_rtv = byte;
1054     2                                  end;
1055     2                          else fcb -> fab$b_rtv = 0;
1056     2                  goto next_opt;
```

```
1057    2
1058    2
1059    2       opt(rewind_close):
1060    2               fcb -> fab$l_fop(fab$v_rwc) = specified & bitval(0);
1061    2               goto next_opt;
1062    2
1063    2
1064    2       opt(rewind_open):
1065    2               fcb -> fab$l_fop(fab$v_rwo) = specified & bitval(0);
1066    2               goto next_opt;
1067    2
1068    2
1069    2       opt(scalarvarying):
1070    2               fcb -> attr(atr_v_scalvar) = specified & bitval(0);
1071    2               goto next_opt;
1072    2
1073    2
1074    2       opt(shared_read):
1075    2               if specified & bitval(0)
1076    2                       then do;
1077    3                               if fcb -> fab$b_shr(fab$v_nil)
1078    3                                       then do;
1079    4                                               error_code = pli$_noshare;
1080    4                                               goto opt_error;
1081    4                                               end;
1082    3                               fcb -> fab$b_shr(fab$v_get) = true;
1083    3                               fcb -> fab$b_shr(fab$v_upi) = true;
1084    3                               end;
1085    2                       else fcb -> fab$b_shr(fab$v_get) = false;
1086    2               goto next_opt;
1087    2
1088    2
1089    2       opt(shared_write):
1090    2               if specified & bitval(0)
1091    2                       then do;
1092    3                               if fcb -> fab$b_shr(fab$v_nil)
1093    3                                       then do;
1094    4                                               error_code = pli$_noshare;
1095    4                                               goto opt_error;
1096    4                                               end;
1097    3                               fcb -> fab$b_shr(fab$v_put) = true;
1098    3                               fcb -> fab$b_shr(fab$v_get) = true;
1099    3                               fcb -> fab$b_shr(fab$v_del) = true;
1100    3                               fcb -> fab$b_shr(fab$v_upd) = true;
1101    3                               fcb -> fab$b_shr(fab$v_upi) = true;
1102    3                               end;
1103    2                       else do;
1104    3                               fcb -> fab$b_shr(fab$v_put) = false;
1105    3                               fcb -> fab$b_shr(fab$v_del) = false;
1106    3                               fcb -> fab$b_shr(fab$v_upd) = false;
1107    3                               end;
1108    2               goto next_opt;
1109    2
1110    2
1111    2       opt(spool):
1112    2               fcb -> fab$l_fop(fab$v_spl) = specified & bitval(0);
1113    2               goto next_opt;
```

```
1114     2
1115     2
1116     2          opt(supersede):
1117     2                  if specified & bitval(0)
1118     2                          then do;
1119     3                                  if fcb -> attr(atr_v_app)
1120     3                                          then do;
1121     4                                                  error_code = pli$_conappsup;
1122     4                                                  goto opt_error;
1123     4                                                  end;
1124     3                                  fcb -> fab$l_fop(fab$v_mxv) = false;
1125     3                                  fcb -> fab$l_fop(fab$v_cif) = false;
1126     3                                  fcb -> fab$l_fop(fab$v_sup) = true;
1127     3                                  fcb -> fab$l_fop(fab$v_nef) = true;
1128     3                                  fcb -> rab$l_rop(rab$v_eof) = false;
1129     3                                  end;
1130     2                          else do;
1131     3                                  if ^fcb -> attr(atr_v_app)
1132     3                                          then do;
1133     4                                                  fcb -> fab$l_fop(fab$v_mxv) = false;
1134     4                                                  fcb -> fab$l_fop(fab$v_cif) = false;
1135     4                                                  fcb -> fab$l_fop(fab$v_sup) = false;
1136     4                                                  fcb -> fab$l_fop(fab$v_nef) = false;
1137     4                                                  fcb -> rab$l_rop(rab$v_eof) = false;
1138     4                                                  end;
1139     3                                  end;
1140     2                  goto next_opt;
1141     2
1142     2
1143     2          opt(system_protection):
1144     2                  longtemp = system_prot;
1145     2                  goto protection;
1146     2
1147     2
1148     2          opt(temporary):
1149     2                  fcb -> fab$l_fop(fab$v_tmp) = specified & bitval(0);
1150     2                  goto next_opt;
1151     2
1152     2
1153     2          opt(truncate):
1154     2                  fcb -> fab$l_fop(fab$v_tef) = specified & bitval(0);
1155     2                  goto next_opt;
1156     2
1157     2
1158     2          opt(world_protection):
1159     2                  longtemp = world_prot;
1160     2                  goto protection;
1161     2
1162     2
1163     2          opt(write_behind):
1164     2                  fcb -> rab$l_rop(rab$v_wbh) = specified & bitval(0);
1165     2                  goto next_opt;
1166     2
1167     2
1168     2          opt(write_check):
1169     2                  fcb -> fab$l_fop(fab$v_wck) = specified & bitval(0);
1170     2                  goto next_opt;
```

```
 1171    2
 1172    2
```

```
1173 !  2       /* utility routines */
1174    2       protection:
1175    2               if specified
1176    2               then
1177    2                   if verify(addrval(0) -> stringtemp,'rwedRWED') ^= 0
1178    2                       then do;
1179    2                               error_code = pli$_invprot;
1180    2                               goto opt_error;
1181    2                               end;
1182    2               if ^specified
1183    2               then do;
1184    2                       prot(longtemp + no_read) = true;
1185    2                       prot(longtemp + no_write) = true;
1186    2                       prot(longtemp + no_execute) = true;
1187    2                       prot(longtemp + no_delete) = true;
1188    2                       end;
1189    2               else do;
1190    2                       if (index(addrval(0) -> stringtemp,'r') = 0 &
1191    2                               index(addrval(0) -> stringtemp,'R') = 0)
1192    2                               then prot(longtemp + no_read) = true;
1193    2                       if (index(addrval(0) -> stringtemp,'w') = 0 &
1194    2                               index(addrval(0) -> stringtemp,'W') = 0)
1195    2                               then prot(longtemp + no_write) = true;
1196    2                       if (index(addrval(0) -> stringtemp,'e') = 0 &
1197    2                               index(addrval(0) -> stringtemp,'E') = 0)
1198    2                               then prot(longtemp + no_execute) = true;
1199    2                       if (index(addrval(0) -> stringtemp,'d') = 0 &
1200    2                               index(addrval(0) -> stringtemp,'D') = 0)
1201    2                               then prot(longtemp + no_delete) = true;
1202    2                       opn.status(protect) = true;
1203    2                       end;
1204    2               goto next_opt;
1205    2
1206 !  2       /* bottom of loop */
1207    2
1208    2       next_opt:
1209    2               if specified
1210    2                   then do;
1211    3                       if openv -> optbit.env_number = 0
1212    3                           then openv = addr(end_opt);
1213    3                       if declared_environment -> optbit.env_number = 0
1214    3                           then declared_environment = addr(end_opt);
1215    3                       if openv -> optbit.env_number =
1216    3                           declared_environment -> optbit.env_number
1217    3                        then do;
1218    4                           call get_opt_val(openv,0);
1219    4                           call get_opt_val(declared_environment,1);
1220    4                           end;
1221    3                        else do;
1222    4                           if openv -> optbit.env_number <
1223    4                                   declared_environment -> optbit.env_number
1224    4                               then call get_opt_val(openv,0);
1225    4                               else call get_opt_val(declared_environment,0);
1226    4                           end;
1227    3                       end;
1228    2               end;
```

```
1229    1               return(1);
1230    1
1231    1
1232    1       get_opt_val: procedure(optpt,valnum);
1233  ! 2       /* parameter declarations */
1234    2       dcl optpt                   pointer;
1235    2       dcl valnum                  fixed bin(7);
1236    2
1237    2       next_specified_env_number = optpt -> optbit.env_number;
1238    2       if next_specified_env_number = 0 ! next_specified_env_number = unused_envir_opt
1239    2               then do;
1240    3                       next_specified_env_number = unused_envir_opt;
1241    3                       return;
1242    3                       end;
1243    2
1244    2       goto    opt_typ(env_type(next_specified_env_number));
1245    2
1246    2       opt_typ(bittyp):
1247    2               bitval(valnum) = optpt -> optbit.bitt;
1248    2               optpt = addr(optpt -> bitnext);
1249    2               if valnum = 1 & bitval(0) ^= bitval(1)
1250    2                       then goto con_opt_exit;
1251    2               return;
1252    2
1253    2       opt_typ(longtyp):
1254    2               longval(valnum) = optpt -> long;
1255    2               optpt = addr(optpt -> longnext);
1256    2               if valnum = 1 & longval(0) ^= longval(1)
1257    2                       then goto con_opt_exit;
1258    2               return;
1259    2
1260    2       opt_typ(quadtyp):
1261    2               quadval(valnum,0) = optpt -> address -> qvalue(0);
1262    2               quadval(valnum,1) = optpt -> address -> qvalue(1);
1263    2               optpt = addr(optpt -> addrnext);
1264    2               if valnum = 1 & (quadval(0,0) ^= quadval(1,0) !
1265    2                               quadval(0,1) ^= quadval(1,1))
1266    2                       then goto con_opt_exit;
1267    2               return;
1268    2
1269    2       opt_typ(stringtyp):
1270    2               addrval(valnum) = addr(optpt -> string);
1271    2               optpt = addr(optpt -> stringnext(optpt -> optstringnext.maxsize));
1272    2               if valnum = 1 & addrval(0) -> stringtemp ^=
1273    2                               addrval(1) -> stringtemp
1274    2                       then goto con_opt_exit;
1275    2               return;
1276    2
1277    2       opt_typ(addrtyp):
1278    2               addrval(valnum) = optpt -> address;
1279    2               optpt = addr(optpt -> addrnext);
1280    2               if valnum = 1 & addrval(0) ^= addrval(1)
1281    2                       then goto con_opt_exit;
1282    2               return;
1283    2
1284    2       con_opt_exit:
1285    2               error_code = pli$_conenvopt;
```

```
 1286    2                goto opt_error;
 1287    2
 1288    2        end get_opt_val;
 1289    1
 1290    1        end pli$$envir;
```

COMMAND LINE

```
PLI/DEBUG=NONE/LIS=LIS$:PLIENVIR/OBJ=OBJ$:PLIENVIR MSRC$:PLIENVIR+LIB$:PL1RTSRC.TLB/LIB
```

PLIDELETE
LIS

PLICONVRT
LIS

PLIDATA
LIS

PLICONTRL
LIS

PLIDATE
LIS

PLICVTPIC
LIS

PLIENVIR
LIS