Appendix E – plib.a library, selected functions

1.0 General Information

1.1 Application Description

The following Postgres-C functions are included in the /rfc_arc/lib/process/plib.a library:

- get_pecrsep - retrieves the 96 value data array from the pecrsep table.
- get_pedpsep - retrieves the 31 value data array from the pedpsep table.
- get_peqpsep - retrieves the 4 value data array from the peqpsep table.
- get_pehpsep - retrieves the 24 value data array from the pehpsep table.

Given the lid, SHEF code and date, these functions return arrays of the data and quality flags from the designated table.

1.2 Design Considerations

The returned arrays are based on the table structures so that you can find the data for a specific time based on its position within the array.

- pecrsep - 96 15 minute values per row for given day
    - array pos - time:
        - 0 - 0000z
        - 1 - 0015z
        - 2 - 0030z
        - 4 - 0100z
        - 24 - 0600z
        - 48 - 1200z
        - 95 - 2345z
- pedpsep - 31 daily values per row for given month
    - array pos - day:
        - 0 - 1st
        - 1 - 2nd
        - 14 - 15th
        - 30 - 31st
- peqpsep - 4 six hour values per row for given day
    - array pos - end time:
        - 0 - 00z
        - 1 - 06z
        - 2 - 12z
        - 3 - 18z
- pehpsep - 24 hourly values per row for given day
    - array pos - end time:
        - 0 - 00z
        - 1 - 01z

- 12 - 12z
- 23 - 23z

2.0 Configuration Information

Your program must include the following:

- The process.h header file (see Attachment A for contents), so that the functions and a needed variable type are defined:
  - #include "/rfc_arc/lib/process/process.h"
- A variable defined using the ID_PED typedef structure from the process.h file, for example:
  - ID_PED idped;
- Defined array variables to hold the data and flags. The length of the array depends on which table you will be accessing; the following variable definitions are an example for the pecrsep table:
  - double val[96];
  - char flag[96][2];
- Open the database:
  - EXEC SQL connect to :dbname;

The makefile needs to include the /rfc_arc/lib/process/plib..a library file (see Attachment C for a sample makefile).

3.0 User How-To

The input arguments are similar for all of the functions (see Attachment A). An example call using get_pecrsep is presented below:

```
err = get_pecrsep (idped, adate, val, flag);
```

where:

- ID_PED idped- contains the lid and SHEF code
  - idped.sid = lid
  - idped.spe1 = first character of SHEF physical element code
  - idped.spe2 = second character of SHEF physical element code
  - idped.sdur = duration code
  - idped.st = type code
  - idped.ss = source code
  - idped.se = extremum code
  - idped.sp = probability code
- char adate[11] - the ansi-formatted date
  - CCYY-MM-DD
- double val[96] - this will contain the returned data values
  - note the array length is dependent on which function you are using
- char flag[96][2] - this will contain the returned quality flags

- note the array length is dependent on which function you are using

The return value of the function is the integer SQLCODE where:

- 0 = successful
- 100 = no rows found
- < 0 = error occurred on query

Attachment B contains sample code using these functions.

Attachment A
process.h

```
#include <pgtypes_date.h>

typedef struct ID_PED {
   char sid[9];
   char spe1[2],spe2[2],sdur[2],st[2],ss[2],se[2],sp[2];
 } ID_PED;

int get_pecrsep(ID_PED,char[11],double[96],char[96][2]);
int get_pedpsep(ID_PED,char[11],double[31],char[31][2]);
int get_peqpsep(ID_PED,char[11],double[4],char[4][2]);
int get_pehpsep(ID_PED,char[11],double[24],char[24][2]);

int chk_sok(ID_PED,int[],char[][2],int);

int get_sysdate(char[2][9],int);
int ystrda(int);
int tmro(int);
int check_date(char[9]);
int ansi2int(char[20],int *,int *);
int int2ansi(int,int,char[2],char[20]);
void inc15min(int *,int *,int);
```

Attachment B
Sample program code

```
/* getdata_sample.pgc
    This program will get the 12z value from each table for:
      lid = ZIOU1
      data type = precipitation
      date = 4/20/2006
*/

#include <stdio.h>
#include <string.h>
#include <stdlib.h>
#include "/rfc_arc/lib/process/process.h"

main()
{
   ID_PED idped;
   char adate[20];
   double rawval[96],dlyval[31],sixval[4],hlyval[24];
   char rawflag[96][2],dlyflag[31][2],sixflag[4][2],hlyflag[24][2];
   int err;
   double rawval12z,dlyval12z,sixval12z,hlyval12z;
   char rawflag12z[2],dlyflag12z[2],sixflag12z[2],hlyflag12z[2];
   EXEC SQL begin declare section;
    char dbname[]="adb_ob7str";
   EXEC SQL end declare section;

   EXEC SQL connect to :dbname;

/* GET DATA FROM PECRSEP TABLE */
   strcpy(idped.sid,"ZIOU1");
   strcpy(idped.spe1,"P");
   strcpy(idped.spe2,"C");
   strcpy(idped.sdur,"I");
   strcpy(idped.st,"R");
   strcpy(idped.ss,"G");
   strcpy(idped.se,"Z");
   strcpy(idped.sp,"Z");
   strcpy(adate,"2006-04-20");

   err = get_pecrsep(idped,adate,rawval,rawflag);
   if (err == 100)
     printf("get_pecrsep: no data found\n");
   else if (err < 0)
     printf("** get_pecrsep: sql error occurred - %d **\n",err);
   else {
     rawval12z = rawval[48];  /* 12z value */
     strcpy(rawflag12z,rawflag[48]);
     printf("pecrsep 12z: %f %s\n",rawval12z,rawflag12z);
   }

/* GET DATA FROM PEDPSEP TABLE */
   strcpy(idped.spe1,"P");
   strcpy(idped.spe2,"P");
   strcpy(idped.sdur,"D");
```

```
   strcpy(idped.st,"1");
   strcpy(adate,"2006-04-01");   /* pedpsep date is always first of month */

   err = get_pedpsep(idped,adate,dlyval,dlyflag);
   if (err == 100)
     printf("get_pedpsep: no data found\n");
   else if (err < 0)
     printf("** get_pedpsep: sql error occurred - %d **\n",err);
   else {
     dlyval12z = dlyval[19];  /* 20th value */
     strcpy(dlyflag12z,dlyflag[19]);
     printf("pedpsep 12z: %f %s\n",dlyval12z,dlyflag12z);
   }

/* GET DATA FROM PEQPSEP TABLE */
   strcpy(idped.sdur,"Q");
   strcpy(adate,"2006-04-20");

   err = get_peqpsep(idped,adate,sixval,sixflag);
   if (err == 100)
     printf("get_peqpsep: no data found\n");
   else if (err < 0)
     printf("** get_peqpsep: sql error occurred - %d **\n",err);
   else {
     sixval12z = sixval[2];   /* 12z value */
     strcpy(sixflag12z,sixflag[2]);
     printf("peqpsep 12z: %f %s\n",sixval12z,sixflag12z);
   }

/* GET DATA FROM PEHPSEP TABLE */
   strcpy(idped.sdur,"H");

   err = get_pehpsep(idped,adate,hlyval,hlyflag);
   if (err == 100)
     printf("get_pehpsep: no data found\n");
   else if (err < 0)
     printf("** get_pehpsep: sql error occurred - %d **\n",err);
   else {
     hlyval12z = hlyval[12];  /* 12z value */
     strcpy(hlyflag12z,hlyflag[12]);
     printf("pehpsep 12z: %f %s\n",hlyval12z,hlyflag12z);
   }

}
```

Attachment C
Sample makefile

```
PGM=getdata_sample

ECPG       = /usr/bin/ecpg
INCLUDES   = -I/usr/include
LIBS       = /rfc_arc/lib/process/plib.a \
               -L/usr/lib -lecpg -lpq -lpgtypes -lcrypt -lpthread -lm

.SUFFIXES: .pgc .c .o

.pgc.c:
      $(ECPG) -t -r no_indicator -c $(INCLUDES) $<

.c.o:
      gcc -c $(INCLUDES) $<

LIST1=  getdata_sample.o

$(PGM): $(LIST1)
      gcc -o $@ \
                $(LIST1) $(LIBS)

clean:
      rm -f $(LIST1)

$(LIST1):  # in case dependencies change
```