UNITED STATES DEPARTMENT OF THE INTERIOR

GEOLOGICAL SURVEY

FENCE

A Computer Program To Create Files of Borehole Data and To Construct Fence Sections of Boreholes Interactively

By

John O. Kork

Open-File Report 81-163

1981

TABLE OF CONTENTS

LIST OF FIGURES AND TABLES

INTRODUCTION

FENCE is a system of computer programs designed to store, retrieve, and display borehole stratigraphic data. Using FENCE a user can create a data base containing location, surface elevation, and unit tops and bottoms for each borehole. As many as twenty-five boreholes can then be displayed in a fence section diagram using any unit present in all the holes as the level datum.

The programs are written in the PL/1 language and designed to run on the USGS Honeywell Multics system in Denver using a Tektronix 4014 graphics terminal and the DISSPLA graphics software package. The programs contain some Multics-dependent operations, but adaption to another system should be relatively easy. Except for one scale-annotation subroutine call to DISSPLA, the graphics calls can be implemented easily on Calcomp or similar software.

COMPONENTS OF FENCE

There are three components in FENCE, a file of unit codes, a keyed PL/1 data base, and a set of programs. The file of unit codes is created by the user via a text editor. These codes are used for checking input to the data base and annotating units on the fence section diagram. The data base is created using interactive programs from FENCE. After the unit-code file and data base have been created, the user can generate fence maps interactively on the graphics terminal.

CREATING THE UNIT-CODE FILE

The unit-code file contains fifteen or fewer lines, each containing a four-letter (or fewer) unit code. The code may consist of letters or numerals in upper or lower case. These codes require that a perfect match be made when inserting information about a unit into the data base. When the codes are displayed, however, the first letter is upper case and the rest are lower case. The unit-code file must reside in a segment named "fence_map_unit_code".

The unit codes must be arranged in stratigraphic order from the top unit to the bottom unit. If fewer than fifteen units are defined, the file can have fewer then fifteen lines. A subset of the unit codes can be present for any given hole, but the units must be in the same stratigraphic order as the unit-code file list. Reversal of units is not allowed.

USING FENCE

To use the FENCE system on the USGS Multics computer in Denver, the user must be able to log in to the system. Upon completion of the normal login sequence, the user can execute FENCE by entering

>udd>Math_ora>JKork>fence_dir>fence

or, if a link is already established, simply

fence. (1)

Once FENCE has been invoked, the program will prompt "Enter task". The user may select any task by choosing one of the five modules. A specified task can be repeated any number of times. Once a task is completed, control is returned to the main program and the user can either choose another task or terminate the session.

The modules named insert and delete are used to edit the data base, and the modules SHOW and PRINT are used to display information from the data base. The module that constructs the FENCE map is called DRAW. The user can exit from FENCE by entering "quit" when prompted by "Enter task". If the user responds with any other word, the program will list the acceptable responses and ask again for the module wanted.

THE DATA BASE

The information for a single borehole record in the data base consists of:

1. a hole id of eight or fewer characters,
2. surface elevation,
3. latitude and longitude in degrees, minutes, and seconds (or decimal degrees),
4. 15 or fewer units consisting of
   a. a unit code
   b. depth to the top of the unit,
   c. depth to the bottom of the unit.

---

(1) All programs are in lower case, but for ease of understanding, well be referred to in all upper case.

Note that for any unit in a hole, the depth to the top of the next lower unit present in the hole equals the depth to the base of the unit. Gaps in the depths are not allowed although units physically missing can be omitted.

CREATING AND EDITING THE DATA BASE

There are two modules for editing the data base, INSERT and DELETE. The procedure for changing an erroneous value in a record is to delete the record and insert the correct information.

When the user invokes the insert module by responding "insert" to the "Enter task" prompt, the program initiates the data entry procedure for a borehole. The program checks the input data for consistency and prints the inserted information for the user to verify. When the hole id is entered, the program checks to see whether a hole with that id exists in the data base. If there is a hole with that id, the program displays "Duplicate id" and asks for another hole id. Ideally, the hole id with incorrect data will have been deleted before this. To indicate that the lowest unit top has been entered, the user enters "TD". The program then asks for the depth to the base of the lowest unit measured and verifies the input. The user is then asked if the data are to be displayed for review. If the user responds "yes", the information for that hole are displayed on the Tektronix screen. The program then asks if the user wants the hole inserted into the data base. If the response is "yes", the data are inserted into the data base; otherwise the information is deleted. Any other answer will cause the program to repeat the question. Once the data for a hole are entered, the program asks whether the data for another hole are to be inserted. The user can insert another hole by entering "yes" or can return to the main program by entering "no". By entering "err" for a unit code, the user causes the information entered for a hole to be cancelled. The user can also return to the main program by entering "end" as a hole id.

If a user wants to delete a hole from the data base, "delete" can be answered to the prompt. The user is then queried for a hole id and asked if the data for the hole are to be reviewed. If there is no hole with that id, the user is prompted again for a hole id. After the hole id is entered, the user is asked whether the information is to be displayed. The user is then asked to verify that the hole is to be deleted. A reply of "yes" to the verification query will cause the data to be deleted, and a reply of "no" will cause the data to be retained. The user can then continue deleting holes or return to the main program. Responding "end" to the hole id prompt returns the user immediately to the main program.

DISPLAYING THE DATA FILE

There are two modules for displaying hole information, SHOW and PRINT. The user enters the appropriate module by responding with the module name to the "Enter task" prompt.

The SHOW module is for displaying a hole record on the graphics terminal. The user is prompted for the hole id and then the information for those units present in the hole is displayed on the screen. The user can then choose another hole for display, return to the main program by responding "no" to the prompt "Another hole?", or show another hole by responding "yes" to the prompt. The user can also return to the main program by entering "end" as a hole id.

If the user wants a line-printer listing of all the holes in the data base, this can be obtained by entering "print" to the "Enter task" prompt. The module does not actually cause the listing to be printed; it merely writes a file into the segment "fence_list". The holes are listed in the standard collating sequence by hole id and six holes are printed per printer page. When the file has been written, the user is automatically returned to the main program. The user can cause this file to be printed by using the Multics command

```
dprint fence_list
```

after entering "quit".

USER-CONTROLLED PARAMETERS

The user can specify up to 25 holes to be plotted in a fence map. These holes are plotted in order from left to right with the plot distances between holes being proportional to the distance between holes calculated as if latitude and longitude were Cartesian coordinates.

The user can specify the x and y dimensions of the plot boundary in plot inches. Although every plot is internally scaled to fill as much of the Tektronix screen as possible, the plot-size input can be used to control the shape of the plot page and also the size of the title, hole-id annotation, and unit identifier relative to the rest of the plot. The total of the margins in the x direction is two inches and in the y direction is three inches. Calls to DISSPLA automatically place the plot window within these margins and write the title. Hence if the user wants the annotation smaller relative to the rest of the plot, the plot size can be increased, keeping the ratio between the x and y sides the same. A convenient starting plot size for

the Tektronix screen is x = 15 inches and y = 11 inches.

FENCL will ask the user to insert a title for each plot made. The input should be followed by a dollar sign ($) and enclosed in quotes. An example title is

"this is a title$".

The title letters are written in upper case.

DRAWING A FENCE MAP

After a data base is created, the user can invoke the module that constructs a fence map by entering "draw" after the "Enter task" prompt. The user will then be asked for the first hole id. The computer displays the id for verification and searches the data base to see whether that id is actually in the data base. If there is no such hole, the program informs the user and asks again for the hole id. If the hole is in the data base, the user is prompted for the next hole id. This process continues until either 25 hole ids have been inserted or the user enters "end" to indicate that no more holes are desired.

The user is then asked to choose sea level, a unit top, or a unit base to be a level datum. If no unit is present in all the holes, the user is required to use sea level as the datum. The fence map is then drawn on the Tektronix screen, and the user is notified that the map is complete by a "beep" sound from the terminal. The user can make a copy of the fence map before proceeding. To proceed, the user must press the carriage return on the keyboard twice. The user then has the option of selecting another level datum for the selected holes or returning to the main program.

LEAVING FENCE

The user terminates a session by responding "quit" to the "Enter task" prompt. All files are closed and the user is returned to the Multics command level.

EXAMPLE

DATA FOR THE EXAMPLE

For the data used in the example, the list of unit codes is Kmg, Kdt, Jmb, Jmpc, and Jmw. There are three holes to be inserted into the data base. The information for the three holes is listed in Table 1.

HOLE 1

id: hole1	elevation:	5000.00
	deg	min	sec
latitude	35	0	0.0
longitude	107	0	0.0
	top depth	base depth
Kmg	200.0	300.0
Kdt	300.0	400.0
Jmb	400.0	600.0
Jmpc	600.0	700.0
Jmw	700.0	1100.0

****

HOLE 2

id: hole2	elevation:	4500.00
	deg	min	sec
latitude	35	20	0.0
longitude	107	45	0.0
	top depth	base depth
Kdt	200.0	400.0
Jmpc	400.0	500.0
Jmw	500.0	800.0

****

HOLE 3

id: hole3	elevation:	5300.00
	deg	min	sec
latitude	35	55	0.0
longitude	107	10	0.0
	top depth	base depth
Kmg	400.0	500.0
Kdt	500.0	700.0
Jmpc	700.0	850.0

****

TABLE 1

The following examples show the user creating the unit-code file, creating the data base, correcting an erroneous hole in the data base, and drawing a fence map. Entries preceeded by "(in)" are entered by the user, and entries preceeded by "(out)" show the responses by the computer.

CREATING THE UNIT-CODE FILE

```
(in ) qedx
(in ) i
(in ) Kmg
(in ) Kdt
(in ) Jmb
(in ) Jmpc
(in ) Jmw
(in ) \f
(in ) w fence_map_unit_code
(in ) q
```

ENTERING THE HOLE INFORMATION (FIRST TWO HOLES)

```
(in ) fence
(out)
(out) Enter task:   (in ) insert
(out)
(out) The no of holes is:          0
(out) Do you want a list of the ids? (y or n):   (in ) n
(out)
(out) Another record? (y or n):   (in ) y
(out)
(out) Insert hole id (<= 8 chars) or end to exit:   (in ) holel
(out) id is:  holel    .  OK? (y or n):  (in ) y
(out)
(out) Insert surface elevation:   (in ) 5000
(out) Insert latitude in deg, min, and sec:   (in ) 35 0 0
(out) Insert longitude in deg,min and sec:   (in ) 107 0 0
(out) id:   holel     elevation:       5000.00
(out)    latitude            35 deg          0 min          0.0 sec
(out)    longitude          107 deg          0 min          0.0 sec
(out) Are these values OK? (y or n):  (in ) y
(out)
(out) Insert top unit code                    : (in ) Kmg
(out) Insert unit depth to top                : (in ) 200
(out)
(out) Kmg           200.00     OK? (y or n):   (in ) y
(out)
```

```
(out) Insert next lower unit code                    : (in ) Kdt
(out) Insert unit depth to top                       : (in ) 300
(out)
(out) Kdt          300.00    OK? (y or n):    (in ) y
(out)
(out) Insert next lower unit code                    : (in ) Jmb
(out) Insert unit depth to top                       : (in ) 400
(out)
(out) Jmb          400.00    OK? (y or n):    (in ) y
(out)
(out) Insert next lower unit code                    : (in ) Jmpc
(out) Insert unit depth to top                       : (in ) 600
(out)
(out) Jmpc         600.00    OK? (y or n):    (in ) y
(out)
(out) Insert next lower unit code                    : (in ) Jmw
(out) Insert unit depth to top                       : (in ) 700
(out)
(out) Jmw          700.00    OK? (y or n):    (in ) y
(out)
(out) Insert next lower unit code                    : (in ) TD
(out) Insert base of lowest unit measured            : (in ) 1100
(out)
(out) Do you want to review this hole? (y or n):   (in ) y
(out)
(out) ****************************************
(out) id:   hole1      elevation:       5000.00
(out)                deg      min      sec
(out) latitude        35        0      0.0
(out) longitude      107        0      0.0
(out)               top depth    base depth
(out) Kmg             200.0         300.0
(out) Kdt             300.0         400.0
(out) Jmb             400.0         600.0
(out) Jmpc            600.0         700.0
(out) Jmw             700.0        1100.0
(out) ****************************************
(out)
(out) Is this record ok? (y or n):   (in ) y
(out)
(out) Another record? (y or n):   (in ) y
(out)
(out) Insert hole id (<= 8 chars) or end to exit:   (in ) hole2
(out) id is:  hole2    .  OK? (y or n):  (in ) y
(out)
(out) Insert surface elevation:   (in ) 4500
(out) Insert latitude in deg, min, and sec:   (in ) 35 20 0
(out) Insert longitude in deg,min and sec:   (in ) 107 45 0
(out) id:   hole2      elevation:       4500.00
(out)    latitude            35 deg          20 min          0.0 sec
(out)    longitude          107 deg          45 min          0.0 sec
(out) Are these values OK? (y or n):  (in ) y
(out)
```

```
(out) Insert top unit code                        : (in ) Kdt
(out) Insert unit depth to top                    : (in ) 200
(out)
(out) Kdt          200.00    OK? (y or n):    (in ) y
(out)
(out) Insert next lower unit code                 : (in ) Jmpc
(out) Insert unit depth to top                    : (in ) 400
(out)
(out) Jmpc         400.00    OK? (y or n):    (in ) y
(out)
(out) Insert next lower unit code                 : (in ) Jmw
(out) Insert unit depth to top                    : (in ) 500
(out)
(out) Jmw          500.00    OK? (y or n):    (in ) y
(out)
(out) Insert next lower unit code                 : (in ) TD
(out) Insert base of lowest unit measured         : (in ) 800
(out)
(out) Do you want to review this hole? (y or n):    (in ) y
(out)
(out) ****************************************
(out) id:   hole2      elevation:       4500.00
(out)                deg       min       sec
(out) latitude        35        20        0.0
(out) longitude      107        45        0.0
(out)              top depth    base depth
(out) Kdt            200.0         400.0
(out) Jmpc           400.0         500.0
(out) Jmw            500.0         800.0
(out) ****************************************
(out)
(out) Is this record ok? (y or n):    (in ) y
(out)
(out) Another record? (y or n):    (in ) n
(out)
(out) Enter task:    (in ) quit
```

ENTERING THE ERRONEOUS HOLE 3
(NOTICE OPERATOR ERRORS)

```
(in ) fence
(out)
(out) Enter task:    (in ) inert
(out)
(out) You must enter insert, delete, show, print, draw, or quit.
(out) Enter task:    (in ) insert
(out)
(out) The no of holes is:          2
(out) Do you want a list of the ids? (y or n):    (in ) y
(out)
```

```
(out) hole1     hole2
(out) Another record? (y or n):   (in ) y
(out)
(out) Insert hole id (<= 8 chars) or end to exit:   (in ) hole1
(out) id is:  hole1   .  OK? (y or n):  (in ) y
(out)
(out) Duplicate id.
(out) Insert hole id (<= 8 chars) or end to exit:   (in ) hole3
(out) id is:  hole3   .  OK? (y or n):  (in ) y
(out)
(out) Insert surface elevation:   (in ) 0
(out) Insert latitude in deg, min, and sec:   (in ) 0 0 0
(out) Insert longitude in deg,min and sec:   (in ) 0 0 0
(out) id:   hole3     elevation:          0.00
(out)    latitude             0 deg          0 min          0.0 sec
(out)    longitude            0 deg          0 min          0.0 sec
(out) Are these values OK? (y or n):  (in ) n
(out)
(out) Insert surface elevation:   (in ) 5300
(out) Insert latitude in deg, min, and sec:   (in ) 35 55 0
(out) Insert longitude in deg,min and sec:   (in ) 107 10 0
(out) id:   hole3     elevation:       5300.00
(out)    latitude            35 deg         55 min          0.0 sec
(out)    longitude          107 deg         10 min          0.0 sec
(out) Are these values OK? (y or n):  (in ) y
(out)
(out) Insert top unit code                    : (in ) Kmg
(out) Insert unit depth to top                : (in ) 400
(out)
(out) Kmg          400.00    OK? (y or n):   (in ) y
(out)
(out) Insert next lower unit code             : (in ) Kdt
(out) Insert unit depth to top                : (in ) 500
(out)
(out) Kdt          500.00    OK? (y or n):   (in ) y
(out)
(out) Insert next lower unit code             : (in ) Kmg
(out)
(out) New unit code not below than last one.
(out) Last valid unit entered was             : Kdt
(out) Insert next lower unit code             : (in ) Jmpc
(out) Insert unit depth to top                : (in ) 0
(out)
(out) Jmpc           0.00    OK? (y or n):   (in ) y
(out)
(out) The new unit is not deeper than the last one.
(out) Enter the correct depth for this unit.
(out) Insert next lower unit code             : (in ) Jmpc
(out) Insert unit depth to top                : (in ) 700
(out)
(out) Jmpc         700.00    OK? (y or n):   (in ) y
(out)
(out) Insert next lower unit code             : (in ) TD
```

```
(out) Insert base of lowest unit measured    : (in ) 0
(out)
(out) Do you want to review this hole? (y or n):   (in ) n
(out)
(out) Is this record ok? (y or n):   (in ) y
(out)
(out) Another record? (y or n):   (in ) n
(out)
(out) Enter task:   (in ) quit
```

REMOVING THE ERRONEOUS HOLE
(HOLE 3)

```
(in ) fence
(out)
(out) Enter task:   (in ) delete
(out)
(out) Another hole? (y or n):   (in ) y
(out)
(out) Insert hole id (<= 8 chars) or end to exit:   (in ) hole3
(out) id is:  hole3   .  OK? (y or n):  (in ) y
(out)
(out) Would you like to review this hole? (y or n):   (in ) y
(out)
(out) ****************************************
(out) id:   hole3     elevation:       5300.00
(out)                 deg      min      sec
(out) latitude         35       55      0.0
(out) longitude       107       10      0.0
(out)               top depth    base depth
(out) Kmg             400.0         500.0
(out) Kdt             500.0         700.0
(out) Jmpc            700.0           0.0
(out) ****************************************
(out)
(out) Do you want to delete this hole? (y or n):   (in ) y
(out)
(out) Another hole? (y or n):   (in ) n
(out)
(out) Enter task:   (in ) quit
```

ENTERING HOLE 3 CORRECTLY

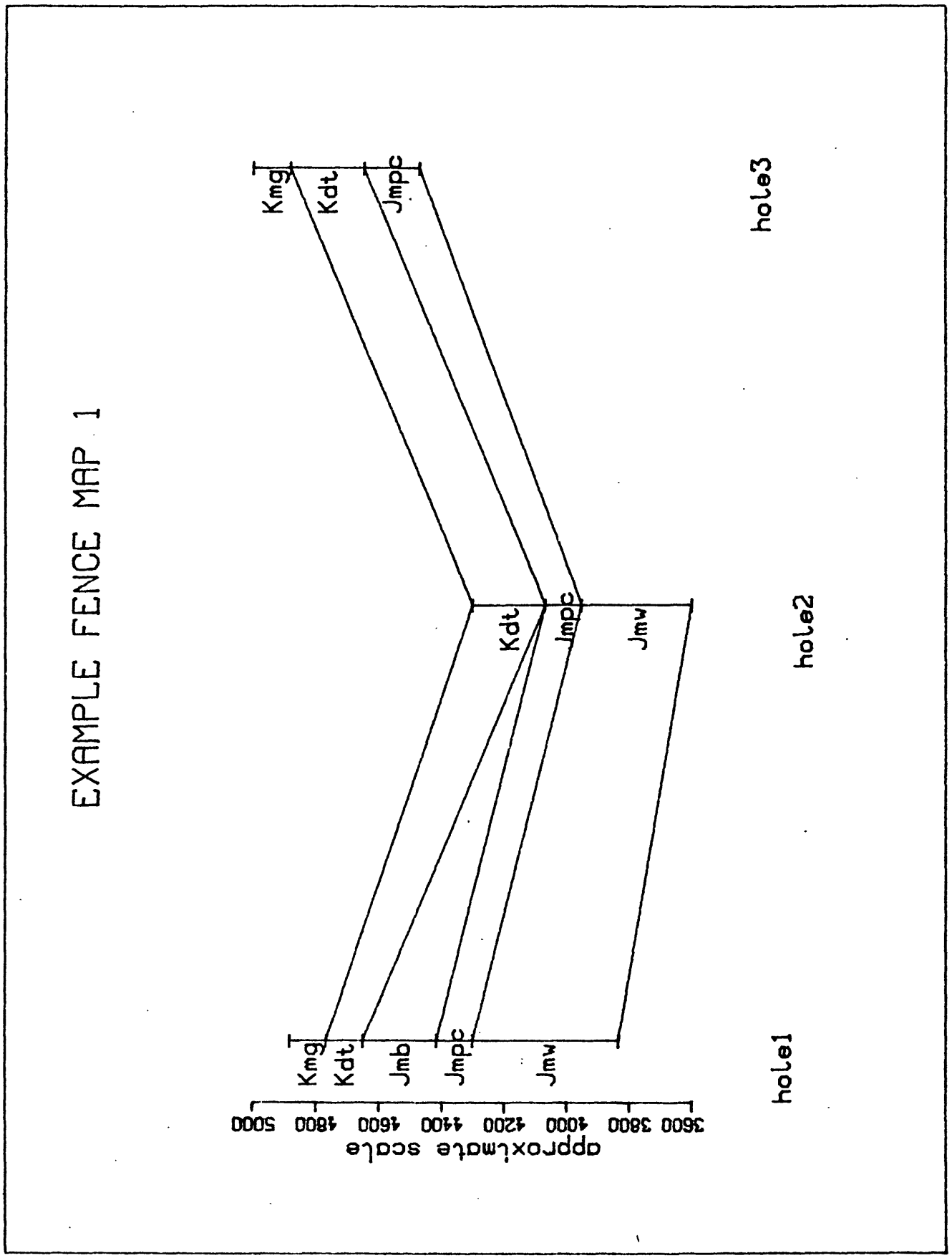
```
(in ) fence
(out)
(out) Enter task:   (in ) insert
(out)
```

```
(out) The no of holes is:          2
(out) Do you want a list of the ids? (y or n):   (in ) n
(out)
(out) Another record? (y or n):   (in ) y
(out)
(out) Insert hole id (<= 8 chars) or end to exit:   (in ) hole3
(out) id is:  hole3   .  OK? (y or n):  (in ) y
(out)
(out) Insert surface elevation:   (in ) 5300
(out) Insert latitude in deg, min, and sec:   (in ) 35 55 0
(out) Insert longitude in deg,min and sec:   (in ) 107 10 0
(out) id:   hole3     elevation:       5300.00
(out)    latitude           35 deg          55 min          0.0 sec
(out)    longitude         107 deg          10 min          0.0 sec
(out) Are these values OK? (y or n):  (in ) y
(out)
(out) Insert top unit code                    : (in ) Kmg
(out) Insert unit depth to top                : (in ) 400
(out)
(out) Kmg          400.00    OK? (y or n):    (in ) y
(out)
(out) Insert next lower unit code             : (in ) Kdt
(out) Insert unit depth to top                : (in ) 500
(out)
(out) Kdt          500.00    OK? (y or n):    (in ) y
(out)
(out) Insert next lower unit code             : (in ) Jmpc
(out) Insert unit depth to top                : (in ) 700
(out)
(out) Jmpc         700.00    OK? (y or n):    (in ) y
(out)
(out) Insert next lower unit code             : (in ) TD
(out) Insert base of lowest unit measured     : (in ) 850
(out)
(out) Do you want to review this hole? (y or n):   (in ) y
(out)
(out) ****************************************
(out) id:   hole3     elevation:       5300.00
(out)                 deg      min      sec
(out) latitude         35       55      0.0
(out) longitude       107       10      0.0
(out)               top depth   base depth
(out) Kmg             400.0        500.0
(out) Kdt             500.0        700.0
(out) Jmpc            700.0        850.0
(out) ****************************************
(out)
(out) Is this record ok? (y or n):   (in ) y
(out)
(out) Another record? (y or n):   (in ) n
(out)
(out) Enter task:   (in ) quit
```

```
(in ) fence
(out)
(out) Enter task:   (in ) draw
(out)
(out) Insert your baud rate (ch/sec) for the Tektronix 4014:
(in ) 120
(out)
(out) Insert x and y page dimensions in inches. (in ) 10 7.5
(out)
(out) xpage =    10.0   ypage =    7.5   OK? (y or n)(in ) y
(out)
(out) Insert hole id (<= 8 chars) or end to exit:   (in ) hole1
(out) id is:  hole1   .  OK? (y or n):  (in ) y
(out)
(out) Insert hole id (<= 8 chars) or end to exit:   (in ) hole2
(out) id is:  hole2   .  OK? (y or n):  (in ) y
(out)
(out) Insert hole id (<= 8 chars) or end to exit:   (in ) hole3
(out) id is:  hole3   .  OK? (y or n):  (in ) y
(out)
(out) Insert hole id (<= 8 chars) or end to exit:   (in ) end
(out)
(out) The reference datum can be:
(out)      1)  elevation
(out)      2)  the top of a unit present in all selected holes
(out)      3)  the base of a unit present in all selected holes
(out)
(out) Insert 1, 2, 3, or 4 to exit:  (in ) 1
(out)
(out) Insert title (<32 char) followed by $.
(out) Multiple word titles must be enclosed in quotes:
(in ) "example fence map 1$"
```

**********FIGURE 1 APPEARS HERE**********

```
(out) The reference datum can be:
(out)      1)  elevation
(out)      2)  the top of a unit present in all selected holes
(out)      3)  the base of a unit present in all selected holes
(out)
(out) Insert 1, 2, 3, or 4 to exit:  (in ) 2
(out)
(out) The units common to all holes are:
(out) Kdt Jmpc
(out) Insert one of these codes:  (in ) Kdt
(out)
(out) Insert title (<32 char) followed by $.
(out) Multiple word titles must be enclosed in quotes:
(in ) "example fence map 2$"
```



Firgure 1. Example Fence Map 1

**********FIGURE 2 APPEARS HERE**********

```
(out) The reference datum can be:
(out)       1)  elevation
(out)       2)  the top of a unit present in all selected holes
(out)       3)  the base of a unit present in all selected holes
(out)
(out) Insert 1, 2, 3, or 4 to exit:  (in ) 4
(out)
(out) Enter task:   (in ) quit
```

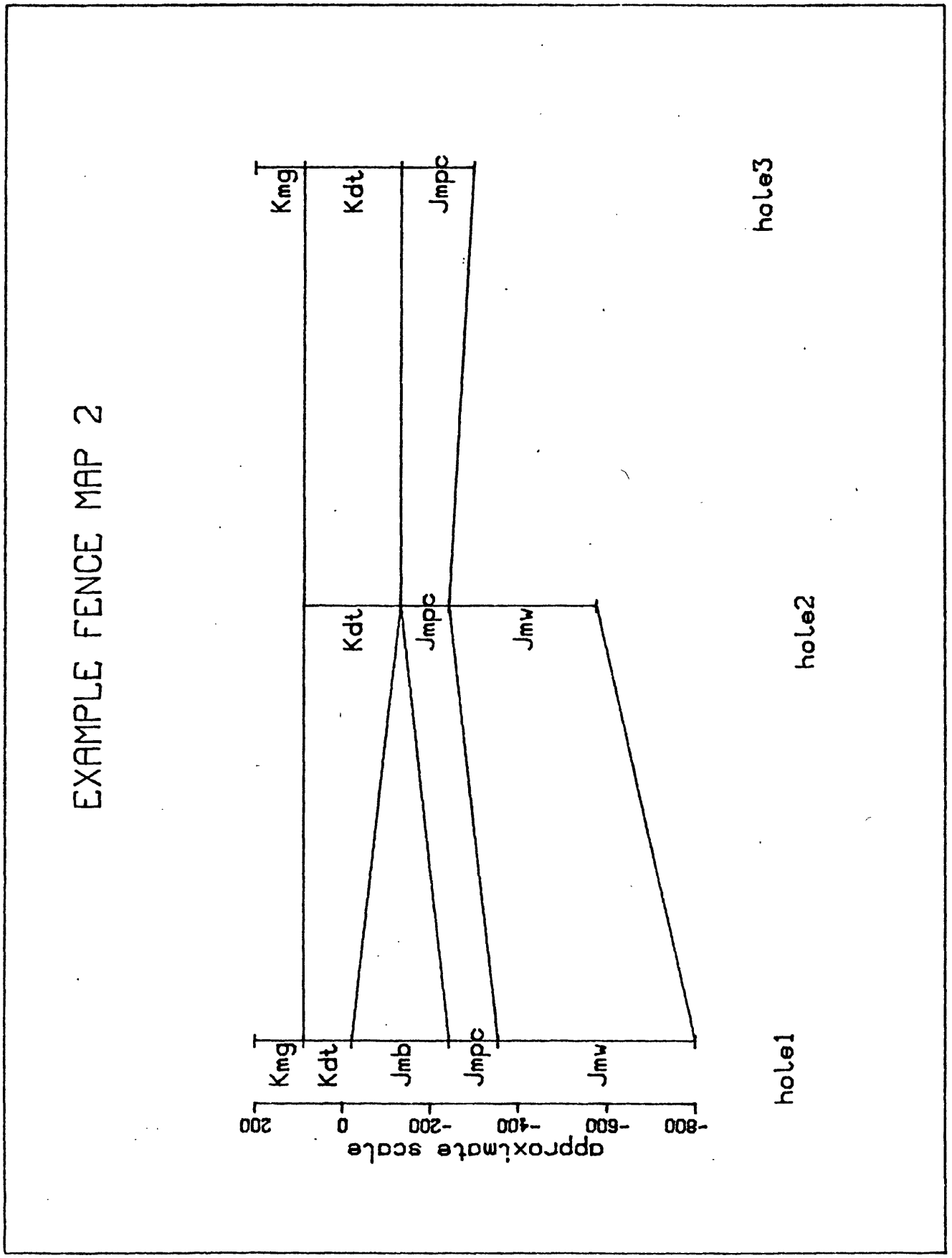


Figure 2. Example Fence Map 2

ANNOTATED PROGRAM LISTING

The subroutine nesting chart in Table 2 shows the order of subroutine calls within the FENCE system. Comments within the programs are preceeded by /* and followed by */.

```
01 fence
  02 exist
  02 pltcde
  02 clear_sysin
  02 inserter
    03 insert_control
      04 flag
      04 char_id
      04 page
      04 unit_control
        05 decode_unit
        05 unit
      04 show_fence_dat
    03 yesno
  02 remove_hole
    03 exist
    03 yesno
    03 char_id
    03 show_fence_dat
  02 display_hole
    03 yesno
    03 char_id
    03 display_hole
  02 list_holes
    03 dp3
  02 draw_fence
    03 plotinfo
      04 clear_sysin
    03 read_posts
      04 clear_sysin
      04 char_id
    03 get_delx
    03 fence_wire
      04 select_ref
        05 clear_sysin
      04 get_datum
        05 clear_sysin
    03 makpic
      04 asr
      04 strtmap
      04 line
      04 maklab
        05 maklabplt
          06 height
          06 messag
      04 holelab
        05 maklabplt
          06 height
          06 messag
      04 boreline
        05 line
      04 drawyaxf
        05 height
          05 yintax
          05 ygraxs
        04 stopit
          05 endpl
        04 dsr
    03 yesno
```

Table 2. Subroutine Nesting Chart

****

****

boreline.pll

```
boreline: proc (post2, x, fmin, scl);
/* procedure to draw a line from the top of the hole
   to the base with tick marks at contacts */
%include post2;
dcl  i fixed bin;
dcl (x, fmin, scl, xlp (2), ylp (2), ytop) float bin;
dcl  line entry options (variable);
/* find the index of the top unit present */
          i = 1;
          do while (unit (i).unit_present = "0"b);
               i = i+1;
          end;
/* draw the top tick marks */
          ytop = scl* (unit (i).topplt-fmin)+1.;
          xlp (1) = x -.05;
          xlp (2) = x+ .05,
          ylp (1) = ytop;
          ylp (2) = ylp (1);
          call line (xlp, ylp, 2, 0);
/* for the units present, draw the tick marks at the
   lower contact */
          do i = 1 to 15;
               if unit (i).unit_present = "1"b then do;
                    ylp (1) = scl* (unit (i).basplt-fmin)+1.;
                    ylp (2) = ylp (1);
                    call line (xlp, ylp, 2, 0);
               end;
          end;
/* draw the vertical line through the hole */
          xlp (1) = x;
          xlp (2) = x;
          ylp (1) = ytop;
          call line (xlp, ylp, 2, 0);
     end;
```

****

****

char_id.pll

```
char_id: proc (charidin);
/* procedure to get an 8 or less character key from
   the user.  a value of "end" causes immediate return */
dcl (sysin, sysprint)file;
dcl  charidin char (8);
dcl  ans char (1);
          do while (ans ^= "y" & charidin ^= "end");
               put skip list
               ("Insert hole id (<= 8 chars) or end to exit:  ");
               get list (charidin);
               if charidin ^= "end" then do;
/* verify the input to be correct */
                    put edit ("id is:  ", charidin,
                         ".  OK? (y or n):  ")
                          (a (8), a (8), a (18));
                    get list (ans);
               end;
          end;
     end;
```

****

****

clear_sysin.pll

```
clear_sysin: proc;
/* system dependent routine to clear the input
   stream after an entry */
dcl (sysin, sysprint)file;
dcl  ch char (1);
dcl (record, transmit)condition;
dcl  io entry options (variable);
          on transmit (sysin) begin;
               close file (sysin);
               call io ("detach", "sysin");
               call io ("attach", "sysin", "syn_", "user_input");
               go to fini;
          end;
          on record (sysin) go to next_char;
```

```
          call io ("detach", "sysin");
          call io ("attach", "sysin", "record_stream_",
                              "user_input");
          open file (sysin) sequential input;
next_char: do while ("1"b);
               read file (sysin) into (ch);
          end;
fini: end;
```

****

****

decode_unit.pl1

```
decode_unit: proc (unit_next_code, i);
/* procedure to decode a four (or less) character input into
   the integer index corresponding to its position in the list of
   possible units in a hole */
dcl (sysin, sysprint) file;
dcl  unit_code (16) char (4) external static;
dcl  i fixed bin;
dcl  unit_next_code char (4);
/* check to see of input code was "err" and set an error flag
   to void this input */
start:    if unit_next_code = "err " then do;
               i = 0;
               return;
          end;
/* check each possible unit code for a match with the input
   code */
loop:     do i = 1 to 16;
/* if a match is found, return with the index i */
               if unit_next_code = unit_code (i) then return;
          end loop;
/* if no match is found, print the list of possible codes and
   prompt for proper input.  input can be aborted by
   inserting "err" */
          put edit ("The code, ", unit_next_code,
               " ,entered is not allowed.")
               (skip, a (10), a (4), a (25));
          put edit ("The allowed codes are:") (skip, a (22));
          put skip;
          i = 1;
          do while (i < 16 & unit_code (i) ^= "flag");
               put edit (unit_code (i)) (x (1), a (4));
               i = i+1;
```

```
          end;
          put edit ("TD  ") (x (1), a (4));
          get list (unit_next_code);
          go to start;
     end;
```

****

****

display_hole.pll

```
display_hole: proc;
/* procedure to display a record from the data base */
dcl (char_id, clear_sysin, show_fence_dat)
     entry options (variable);
dcl  yesno entry (char (60) varying, char (1));
dcl (sysin, sysprint) file;
dcl  fence_dat file;
dcl  answer char (1);
dcl  charidin char (8);
%include postl;
dcl (endfile, error, key) condition;
/* if no record matches the key, start over */
          on key (fence_dat) begin;
               put skip list ("No such hole.  Try again.");
               go to gtkey;
          end,
          open file (fence_dat) keyed sequential update;
gtkey:    answer = "y";
          do while (answer = "y");
/* see if more displays are wanted */
               call yesno ("Another hole? (y or n):  ", answer);
               if answer = "y" then do;
/* get key and display hole */
                    call char_id (charidin);
                    read file (fence_dat) key (charidin)
                         into (post);
                    call show_fence_dat (post);
               end,
          end;
fini:     close file (fence_dat);
     end;
```

****

****

dp3.pl1

```
dp3: proc (key1, post1, key2, post2, key3, post3, nout, nmod);
/* procedure to write scratch files to allow printing
   three borehole records per row of output */
dcl (key1, key2, key3) char (8);
dcl  unit_code (16) char (4) external static;
dcl (nout, nmod, itmp, i, lout, imod) fixed bin;
dcl (l1, l2, l3)fixed bin;
dcl (fence_list, scr1, scr2, scr3) file;
%include post1;
%include post31;
%include post32;
%include post33;
dcl (lin1 (30), lin2 (30), lin3 (30)) char (40) varying;
dcl  endfile condition;
dcl (max, mod) builtin;
          on endfile (scr1) go to labscr2;
          on endfile (scr2) go to labscr3;
          on endfile (scr3) go to putout;
/*  define hole record default values */
          do itmp = 1 to 30;
               lin1 (itmp) = " ";
               lin2 (itmp) = " ";
               lin3 (itmp) = " ";
          end;
          if nout >= 1 then do;
/* write the leftmost record of three to scratch file scr1 */
               post = post1;
               open file (scr1)stream output;
               put file (scr1) edit ((40)"*") (skip, a (40));
               put file (scr1) edit ("id:  ", charid,
                    "elevation:  ", elevation)
                    (skip, a (6), a (8), x (2), a (11),
                    f (12, 2));
               put file (scr1) edit ("deg", "min", "sec")
                    (skip, x (14), a (3), (2) (x (5), a (3)));
               put file (scr1) edit ("latitude", latitude (1),
                    latitude (2), latitude (3))
                    (skip, a (8), x (3), (2) (f (6, 0), x (2)),
                    f (6, 1));
               put file (scr1) edit ("longitude", longitude (1),
                    longitude (2), longitude (3))
                    (skip, a (9), x (2), (2) (f (6, 0), x (2)),
                    f (6, 1));
```

```
/* print the unit data only for those units present in the
   hole */
                put file (scr1) edit ("top depth", "base depth")
                     (skip, x (12), a (9), x (3), a (10));
                do i = 1 to 15;
                     if unit (i).unit_present = "1"b then do;
                          put file (scr1) edit (unit_code (i),
                               unit (i).untdptp, unit (i).untdptb)
                               (skip, a (4), x (6), f (9, 1),
                               x (3), f (9, 1));
                     end;
                end;
                put file (scr1) edit ((40)"*") (skip, a (40));
                put file (scr1) skip;
                close file (scr1);
           end;
           if nout >= 2 then do;
/* write the middle record of three to scratch file scr2 */
                post = post2;
                open file (scr2)stream output,
                put file (scr2) edit ((40)"*") (skip, a (40));
                put file (scr2) edit ("id:  ", charid,
                     "elevation:  ", elevation)
                     (skip, a (6), a (8), x (2), a (11),
                     f (12, 2));
                put file (scr2) edit ("deg", "min", "sec")
                     (skip, x (14), a (3), (2) (x (5), a (3)));
                put file (scr2) edit ("latitude", latitude (1),
                     latitude (2), latitude (3))
                     (skip, a (8), x (3), (2) (f (6, 0), x (2)),
                     f (6, 1));
                put file (scr2) edit ("longitude", longitude (1),
                     longitude (2), longitude (3))
                     (skip, a (9), x (2), (2) (f (6, 0), x (2)),
                     f (6, 1));
/* print the unit data only for those units present
   in the hole */
                put file (scr2) edit ("top depth", "base depth")
                     (skip, x (12), a (9), x (3), a (10));
                do i = 1 to 15;
                     if unit (i).unit_present = "1"b then do;
                          put file (scr2) edit (unit_code (i),
                               unit (i).untdptp, unit (i).untdptb)
                               (skip, a (4), x (6), f (9, 1),
                               x (3), f (9, 1));
                     end;
                end;
                put file (scr2) edit ((40)"*") (skip, a (40));
                put file (scr2) skip;
                close file (scr2);
           end;
           if nout >= 3 then do;
/* write the rightmost record of three to scratch file scr3 */
```

```
                post = post3;
                open file (scr3)stream output;
                put file (scr3) edit ((40)"*") (skip, a (40));
                put file (scr3) edit ("id:  ", charid,
                     "elevation:  ", elevation)
                     (skip, a (6), a (8), x (2), a (11),
                     f (12, 2));
                put file (scr3) edit ("deg", "min", "sec")
                     (skip, x (14), a (3), (2) (x (5), a (3)));
                put file (scr3) edit ("latitude", latitude (1),
                     latitude (2), latitude (3))
                     (skip, a (8), x (3), (2) (f (6, 0), x (2)),
                     f (6, 1));
                put file (scr3) edit ("longitude", longitude (1),
                     longitude (2), longitude (3))
                     (skip, a (9), x (2), (2) (f (6, 0), x (2)),
                     f (6, 1));
/* print the unit data only for those units present
   in the hole */
                put file (scr3) edit ("top depth", "base depth")
                     (skip, x (12), a (9), x (3), a (10));
                do i = 1 to 15;
                     if unit (i).unit_present = "1"b then do;
                          put file (scr3) edit (unit_code (i),
                               unit (i).untdptp, unit (i).untdptb)
                               (skip, a (4), x (6), f (9, 1),
                               x (3), f (9, 1));
                     end;
                end;
                put file (scr3) edit ((40)"*") (skip, a (40));
                put file (scr3) skip;
                close file (scr3);
           end;
/* compile output in full 120 character lines */
getlin:    if nout >= 1 then do;
                l2 = 1;
                l3 = 1;
                open file (scr1) environment (stringvalue)
                     record input title ("record_stream_ -target
                     vfile_ scr1");
                l1 = 1;
/* read the left third of each print line */
                do while ("1"b);
                     read file (scr1) into (lin1 (l1));
                     l1 = l1+1;
                end;
labscr2:        close file (scr1);
           end;
/* read the middle third of each print line */
           if nout >= 2 then do;
                open file (scr2) environment (stringvalue)
                     record input title ("record_stream_ -target
                     vfile_ scr2");
```

```
                l2 = 1;
                do while ("1"b);
                     read file (scr2) into (lin2 (l2));
                     l2 = l2+1;
                end;
labscr3:        close file (scr2);
          end;
/* read the right third of each print line */
          if nout >= 3 then do;
                open file (scr3) environment (stringvalue)
                     record input title ("record_stream_ -target
                     vfile_ scr3");
                l3 = 1;
                do while ("1"b);
                     read file (scr3) into (lin3 (l3));
                     l3 = l3+1;
                end;
putout:         close file (scr3);
          end;
          lout = max (l1, l2, l3)-1;
          imod = mod (nmod, 2);
/* start a new page after six borehole records have
   been written */
          if imod = 1 then put file (fence_list) page;
/* write the full lines for three borehole records */
          else put file (fence_list) skip (3);
          do itmp = 2 to lout;
                put file (fence_list) skip edit (lin1 (itmp),
                     " | ", lin2 (itmp), " | ", lin3 (itmp))
                     (a (40), a (3), a (40), a (3), a (40));
          end;
     end;
```

****

****

draw_fence.pl1

```
draw_fence: proc;
/* main control program for making a fence map */
dcl  sysprint file;
%include post;
dcl (ibaud, ihole, idatum) fixed bin;
dcl (xpage, ypage, dxt (25), fmax, fmin, range, scale) float bin;
dcl  answer char (1);
dcl (again, abort, repeat) bit (1);
```

```
dcl (plotinfo, read_posts, get_delx,
     select_ref, sget_datum, fence_wire,
     makpic, yesno)
     entry options (variable);
/* get plotter number and page size in inches */
          again = "1"b;
          do while (again);
               repeat = "1"b;
               call plotinfo (ibaud, xpage, ypage);
/* read the records for the holes to be included */
               call read_posts (post, ihole, abort);
               if ^abort then
/* calculate the relative distance between successive holes */
                    call get_delx (post, ihole, xpage, dxt,
                         abort);
               do while (repeat & ^abort);
                    if ^abort then
/* calculate the adjusted heights of the contacts */
                         call fence_wire (post, ihole, fmax,
                              fmin, abort);
                    if ^abort then do;
/* calculate the appropriate scale */
                         range = fmax-fmin;
                         if range > 0. then do;
                              scale = (ypage-4.)/ (fmax-fmin);
/* draw the fence map */
                              call makpic (post, ihole, fmax,
                                   fmin, scale, dxt, ibaud,
                                   xpage, ypage),
                         end;
                    end;
                    else if ^abort then do;
                         call yesno
                        ("Do you want another datum? (y or n):  ",
                              answer);
                         if answer = "n" then repeat = "0"b;
                    end;
               end;
               if ^abort then do;
                    call yesno
                   ("Do you want another fence map? (y or n):  ",
                         answer);
                    if answer = "n" then again = "0"b;
               end;
               else do;
                    again = "0"b;
               end;
          end;
     end;
```

```
*************************************************************************
-------------------------------------------------------------------------
*************************************************************************
```

drawyaxf.pl1

```
drawyaxf: proc (fmin, fmax, ypage);
/* procedure to place a relative scale in feet at
   the left edge of the fence map. */
dcl (ypage, fmin, fmax, yorig, ymax, yax, fhite, zero, one)
    float bin;
dcl  hund fixed bin;
dcl (height, yintax, ygraxs) entry options (variable);
          yorig = fmin;
          ymax = fmax;
          yax = ypage-4.;
          fhite = .14;
          zero = 0.;
          one = 1.;
          hund = 100;
          call height (fhite);
          call yintax;
          call ygraxs (yorig, "scale", ymax, yax,
               "approximate scale$", hund, zero, one);
          return;
     end;
```

```
*************************************************************************
-------------------------------------------------------------------------
*************************************************************************
```

exist.pl1

```
exist: proc (name, error);
/* very system dependent routine to check for the existence
   of a segment or directory.  no such file returns "0"b,
   otherwise returns "1"b */
dcl  com_err_ entry options (variable);
dcl  expand_pathname_ entry (char (*), char (*), char (*),
     fixed bin (35));
dcl  hcs_$status_minf entry (char (*), char (*), fixed bin (1),
     fixed bin (2), fixed bin (24), fixed bin (35));
dcl  error_table_$noentry ext fixed bin (35);
```

```
dcl  error_table_$no_dir ext fixed bin (35);
dcl  name char (32),
     dir char (168),
     ename char (32),
     code fixed bin (35),
     type fixed bin (2),
     bit_count fixed bin (24),
     error bit (35);

          call expand_pathname_ (name, dir, ename, code);
          if (code ^= 0) then go to err;

          call hcs_$status_minf (dir, ename, 0, type, bit_count,
               code);
          if code ^= 0 then do;
               if (code = error_table_$noentry)|
                    (code = error_table_$no_dir) then go to false;
               else go to err;
          end;

          if (type = 1)| ((type = 2)& (bit_count ^= 0)) then
               go to true;

false:    error = "0"b,
          go to fini;

true:     error = "1"b;
          go to fini;

err:      call com_err_ (code, "exist");
          go to false;

fini:     return;
     end exist;
```

****

****

fence.pll

```
fence: proc;
/* main driver for program to draw stratigraphic maps */
dcl  name char (32) init ("fence_map_unit_code");
dcl  unit_code (16) char (4) external static;
dcl  labplt (15) char (6) external static;
dcl  ans char (4);
```

```
dcl  i fixed bin;
dcl (ok, code) bit (1);
dcl (exist, clear_sysin, inserter, display_hole, remove_hole,
     draw_fence, list_holes, ec)
     entry options (variable);
dcl  pltcde entry (char (4), char (6));
dcl  endfile condition;
dcl (sysin, sysprint, fence_map_unit_code, fence_dat)file;
          on endfile (fence_map_unit_code) go to start;
          ok = "1"b;
/* check for existence of unit code file */
          call exist (name, code);
/* if unit code file doesn't exist, print error message
   and quit */
          if code = "0"b then do;
               put skip list
                    ("The unit code file does not exist.");
               put skip list
    ("You must create a unit code file before executing fence.");
               put skip list
                  ("See page 2 of the user's manual for help.");
               ok = "0"b;
          end;
/* if unit_code file exists, read it */
          else do;
               unit_code (16) = "TD";
               do i = 1 to 15;
                    unit_code (i) = "flag";
               end;
               open file (fence_map_unit_code) input;
               do i = 1 to 15;
                    get file (fence_map_unit_code) list
                         (unit_code (i)) ;
                    call pltcde (unit_code (i), labplt (i));
               end;
          end;
start:    close file (fence_map_unit_code);
/* request task choice and call appropriate task subroutine */
          do while (ok);
               put skip list ("Enter task:  ");
               get list (ans);
               call clear_sysin;
               if ans = "inse" then call inserter;
               else if ans = "dele" then call remove_hole;
               else if ans = "show" then call display_hole;
               else if ans = "prin" then call list_holes;
               else if ans = "draw" then call draw_fence;
/* if task chosen is to quit, set continuation flag to false */
               else if ans = "quit" then ok = "0"b;
/* if user input not acceptable, print the acceptable inputs
   and ask again for task choice */
               else put skip list
  ("You must enter insert, delete, show, print, draw, or quit.");
```

```
          end;
     end;
```

****

****

fence_wire.pll

```
fence_wire: proc (post, ihole, fmax, fmin, abort);
/* procedure to calculate the heights of the contacts
   adjusted for the reference datum selected */
dcl (sysin, sysprint) file;
%include post;
dcl (fmax, fmin, datum) float bin;
dcl (ihole, i, j, idatum) fixed bin;
dcl  refans char (1);
dcl (again, abort) bit (1);
dcl (select_ref, get_datum) entry options (variable);
dcl (min, max) builtin;
/* select the type of datum to be used (none, unit top
   or base) */
          again = "0"b;
          do while (^again);
               again = "1"b;
               call select_ref (refans);
               if refans = "4" then do;
                    abort = "1"b;
                    return;
               end;
               if refans = "2" | refans = "3" then
/* select the unit to be used for the level reference */
                    call get_datum (post, ihole, idatum, again);
          end;

          fmax = -999999.;
          fmin = -fmax;
/* get the adjustment value for the selected datum */
          do i = 1 to ihole;
               if refans = "1" then datum = post (i).elevation;
               else if refans = "2" then datum =
                    post (i).unit (idatum).untdptp;
               else if refans = "3" then datum =
                    post (i).unit (idatum).untdptb;
               do j = 1 to 16;
/* calculate the adjusted heights for units present in posts */
                    if post (i).unit (j).unit_present = "1"b then
```

```
                        do;
                        post (i).unit (j).topplt =
                             datum-post (i).unit (j).untdptp;
                        post (i).unit (j).basplt =
                             datum - post (i).unit (j).untdptb;
                        fmax = max
                             (fmax, post (i).unit (j).topplt);
                        fmin = min
                             (fmin, post (i).unit (j).basplt);
                   end;
              end;
         end;
    end;
```

****

****

flag.pll

```
flag: proc (post);
/* procedure to initialize all values of a new record to default
   values */
%include post1;
dcl  i fixed bin;
/* initialize page values common to all units in the hole */
          charid = "abcdefgh";
          elevation = -999.;
/* initialize each unit */
          do i = 1 to 4;
               latitude (i) = -999.;
               longitude (i) = -999.;
          end;
          do i = 1 to 16;
               unit_present (i) = "0"b;
               unit (i).untdptp = -999.;
               unit (i).untdptb = -999.;
               unit (i).topplt = -999.;
               unit (i).basplt = -999.;
          end;
    end;
```

****

****

```
get_datum: proc (post, ihole, idatum, again);
/* program to choose the unit to be used as the level reference
   datum */
dcl (ihole, idatum, nkodatum, kodatum (16), i, j) fixed bin;
dcl  codin char (4);
dcl  unit_code (16) char (4) external static;
dcl (ldatum (16), again) bit (1);
%include post;
dcl  clear_sysin entry options (variable);
dcl (sysin, sysprint) file;
/* see which units are present in all the posts */
          nkodatum = 0;
          do i = 1 to 15;
               ldatum (i) = "1"b;
               do j = 1 to ihole;
                    ldatum (i) = ldatum (i) &
                         post (j).unit (i).unit_present;
               end;
/* save the list of unit indices present in all posts
   and also the number of such units */
               if ldatum (i) = "1"b then begin;
                    nkodatum = nkodatum+1;
                    kodatum (nkodatum) = i;
               end;
          end;
/* case for which no unit is present in all holes */
knot:     if nkodatum = 0 then begin;
/* print message that there are no acceptable units and set flag
   to indicate that the reference datum must be selected again */
               put edit
                    ("There are no units common to all holes.")
                    (skip, a (39));
               again = "0"b;
          end;
/* case for which there is at least on unit present in
   all posts */
          else begin;
/* print the list of acceptable units */
               put edit ("The units common to all holes are:",
                    (unit_code (kodatum (i))
                    do i = 1 to nkodatum))
                    (skip, a (34), skip, (16) (a (4)));
               put edit ("Insert one of these codes:  ")
                    (skip, a (28));
/* get unit code for reference datum unit and check to see that
   it is an acceptable unit */
```

```
                get list (codin);
                call clear_sysin;
                do i = 1 to nkodatum;
                     if codin = unit_code (kodatum (i)) then
                          begin;
                          idatum = kodatum (i);
                          again = "1"b;
                          return;
                     end;
                end;
                put edit ("You misspelled the code.  Try again.")
                     (skip, a (35));
                go to knot;
          end;
     end;
```

****

****

get_delx.pll

```
get_delx: proc (post, ihole, xpage, dxt, abort);
/* procedure to calculate the relative distance between
   successive posts in plot inches */
%include post;
dcl (ihole, i) fixed bin;
dcl (dxt (25), sum, xpage) float bin;
dcl  abort bit (1);
dcl  sqrt builtin;

          sum = 0.;
          dxt (1) = 0.;
/* accumulate sum of all distances */
          do i = 2 to ihole;
/* calculate the distance using latitude and longitude as if
   they were cartesian coordinates */
               dxt (i) = sqrt ((post (i).latitude (4)-
                    post (i-1).latitude (4))**2
                    + (post (i).longitude (4)-
                    post (i-1).longitude (4))**2);
               sum = sum+dxt (i);
          end;
/* scale the distances so that the x axis with be three
   inches less than the page size */
          do i = 2 to ihole;
               dxt (i) = (xpage-3.)*dxt (i)/sum;
```

```
        end;

    end;
```

****

****

holelab.pll

```
holelab: proc (x, id, jj);
/* procedure to plot the hole id (8 characters or less) */
dcl  id char (8);
dcl (x, ht, xp, yp)float bin;
dcl  jj fixed bin;
dcl  mod builtin,
dcl  maklabplt entry options (variable);
/* set height in inches */
          ht = .14;
/* start the label as if it were always 8 characters long */
          xp = x-3.5*ht;
/* calculate vertical location to alternate between two levels to
   help avoid overprinting */
          yp = .4-mod (jj, 3)* (ht+.03);
          call maklabplt (xp, yp, id, 8, ht);
    end;
```

****

****

insert_control.pll

```
insert_control: proc (charidin, post, hole_ok, abort);
/* procedure to control the input of data for one record */
dcl (flag, char_id, pagin, unit_control, show_fence_dat,
     clear_sysin)
     entry options (variable);
dcl  yesno entry (char (60) varying, char (1));
dcl (sysin, sysprint, fence_dat) file;
dcl  charidin char (8);
```

```
dcl  answer char (1);
%include post1;
dcl (abort, hole_ok, insert_ok) bit (1);
dcl  key condition;
/* condition to allow insertion only if no record presently in
   the data base has the same key */
          on key (fence_dat) begin;
               insert_ok = "1"b;
               go to ins;
          end;
/* initialize all values for the record */
          hole_ok = "0"b;
          do while (^hole_ok);
               insert_ok = "0"b;
               call flag (post);
/* get the key for the record to be inserted */
               call char_id (charidin);
               post.charid = charidin,
               if charidin ^= "end" then do;
/* check to insure that the key has not been used yet */
                    read file (fence_dat) key (charidin)
                         into (post);
                    put skip list ("Duplicate id.");
/* get page header information */
ins:                do while (insert_ok);
                         insert_ok = "0"b;
                         call pagin (post),
/* get information for all units */
                         call unit_control (post, abort);
/* abort program run on error return with hole information */
                         if ^abort then do;
                              call yesno
        ("Do you want to review this hole? (y or n):  ", answer);
                              if answer = "y" then
                                   call show_fence_dat (post);
                              call yesno
                        ("Is this record ok? (y or n):  ", answer);
                              if answer = "y" then
                                   hole_ok = "1"b;
                         end;
                    end;
               end;
               else do;
                    hole_ok = "1"b;
                    abort = "1"b;
               end;
          end;
     end;
```

```
*************************************************************************
-------------------------------------------------------------------------
*************************************************************************
```

inserter.pll

```
inserter: proc;
/* driver for inserting data for a hole into the data base */
dcl  insert_control entry options (variable);
dcl  exist entry options (variable);
dcl  yesno entry (char (60) varying, char (1));
dcl (sysin, sysprint) file;
dcl  fence_dat file;
dcl  answer char (1);
dcl  charidin char (8);
dcl (rec_cnt, i) fixed bin;
dcl  charidout (1000) character (8);
%include post1;
dcl (abort, next_ok, hole_ok, code) bit (1);
dcl  name char (32) init ("fence_dat");
dcl (endfile, error, key) condition;
/* open the data file, count the number of records and save
   the keys */
/* file opening to define the input file the first time around */
          on error go to closeout;
          call exist (name, code);
          if ^code then do;
               open file (fence_dat) keyed sequential update;
               write file (fence_dat) key
                    from (charidin)
                    from (post);
               delete file (fence_dat) key (charidin);
               close file (fence_dat);
          end;
start:    rec_cnt = 0;
          open file (fence_dat) keyed sequential input;
          on endfile (fence_dat) go to counted;
          do while ("1"b);
               abort = "0"b;
               read file (fence_dat) keyto (charidin)
                    into (post);
               rec_cnt = rec_cnt+1;
               charidout (rec_cnt) = charidin;
          end;
/* print the number of holes and, if requested, the list
   of keys */
```

```
counted:  put edit ("The no of holes is:  ", rec_cnt)
               (skip, a (21), f (8));
          call yesno
          ("Do you want a list of the ids? (y or n):  ", answer);
          if answer = "y" then
               put edit ((charidout (i) do i = 1 to rec_cnt))
               (skip, (10) (a (8), x (2)));
          close file (fence_dat);
          open file (fence_dat) keyed update;
/* see whether another hole is to be inserted */
          next_ok = "1"b;
          do while (next_ok);
               call yesno ("Another record? (y or n):  ", answer);
               if answer = "y" then do;
/* get the information for a new record */
                    call insert_control (charidin, post, hole_ok,
                         abort);
                    charidin = post.charid;
                    if hole_ok & ^abort then
/* write the record into the data base */
                         write file (fence_dat) keyfrom
                              (charidin) from (post);
               end;
               else next_ok = "0"b;
          end;
closeout: close file (fence_dat);
     end;
```

****

****

line.pll

```
line: proc (x, y, n, i);
/* procedure to draw a line from x(1),y(1)
   to x(2),y(2) where the coordinates are in
   plot inches */
dcl (x (2), y (2)) float bin;
dcl (n, i) fixed bin;
dcl  curve entry options (variable);
          call curve (x, y, n, i);
          return;
     end;
```

```
*****************************************************************
-----------------------------------------------------------------
*****************************************************************
```

list_holes.pll

```
list_holes: proc;
/* procedure controling the printer output of the whole
   data base */
dcl (key1, key2, key3) char (8);
dcl (nout, nmod) fixed bin;
dcl  more_ok bit (1);
dcl (fence_dat, fence_list) file;
%include post31;
%include post32;
%include post33;
dcl  dp3 entry options (variable);
dcl  endfile condition;
/* at end of file, set loop flag and make last
   call to write the to output file */
          on endfile (fence_dat) begin;
               more_ok = "0"b;
               go to wrt;
          end;
          open file (fence_dat) keyed sequential input;
          more_ok = "1"b;
/* read the records three at a time and
   then write to output file */
          nmod = 0;
          do while (more_ok);
               nout = 0;
               read file (fence_dat) keyto (key1) into (post1);
               nout = 1;
               read file (fence_dat) keyto (key2) into (post2);
               nout = 2;
               read file (fence_dat) keyto (key3) into (post3);
               nout = 3;
/* increment counter for printing on top of page
   or bottom of page */
wrt:           nmod = nmod+1;
               if nout > 0 then do;
                    call dp3 (key1, post1, key2, post2, key3,
                         post3, nout, nmod);
               end;
          end;
/* close files */
          close file (fence_dat);
          close file (fence_list);
     end;
```

****

****

maklab.pl1

```
maklab: proc (post2, x, fmin, scl, labmem);
/* procedure to place labels for units */
%include post2;
dcl (x, xp, yp, yl, yu, scl, ht, fmin)float bin;
dcl  i fixed bin;
dcl  labmem (16)bit (1);
dcl  maklabplt entry options (variable);
dcl  labplt (15) char (6) external static;
          do i = 1 to 15;
/*  next line is just a trial to see of notation is better
   when included for all units present in every hole */
               labmem = "0"b;
/* set initial character height to .14 inches */
               ht = .14;
/* if the unit is present and has not already been labeled by a
   connecting hole to the left, make the label */
               if unit (i).unit_present = "1"b &
                    labmem (i) = "0"b then do;
                    yu = scl* (unit (i).topplt-fmin)+1.;
                    yl = scl* (unit (i).basplt-fmin)+1.;
/* if the thickness of the unit at the present hole is less
   than the default character height, calculate a height
   that can be plotted with the unit thickness */
                    if yu-yl < ht then ht = yu-yl;
                    xp = x-ht*2.5;
                    yp = (yl+yu-ht)/2.;
/* plot the label */
                    call maklabplt (xp, yp, labplt (i), 6, ht);
                    labmem (i) = "1"b;
               end;
/* These commands will change the annotation so that
   unit labels are not drawn if there is a label for that
   unit on a previously drawn post
   else if unit (i) .untdptb <= unit (i).untdptp
   then labmem (i) = "0"b;
   */
          end;
     end;
```

****

****

maklabplt.pll

```
maklabplt: proc (xc, yc, lab, nchar, ht);
/* procedure to interface to disspla package for message
   plotting */
dcl (xc, yc, ht) float bin;
dcl  nchar fixed bin;
dcl  lab char (6);
dcl (messag, height) entry options (variable),
          call height (ht);
          call messag (lab, nchar, xc, yc);
          return;
     end;
```

****

****

makpic.pll

```
makpic: proc (post, inole, fmax, fmin, scl, dxt, ibaud, xpage,
    ypage);
/* procedure to control the actual calculations and plotting of
   the fence map */
%include post;
dcl (ihole, ibaud, jj, ilow, itopblnk, ibase (2), itop (2), i)
    fixed bin;
dcl (fmax, fmin, scl, xt (2), yt (2), dxt (25), xpage, ypage)
    float bin;
dcl (asr, strtmap, line, maklab, holelab,
     stopit, dsr, boreline, drawyaxf)
     entry options (variable);
dcl (labmem (16), drawok) bit (1);
dcl  max builtin;
          call asr (">iml>disspla", "-after", "working_dir");
          call strtmap (ibaud, xpage, ypage);
/* labmem indicates whether a unit has been annotated at
   a hole located further left on the plot--initialize labmem
   before the program gets into the plotting loop */
          do i = 1 to 15;
               labmem (i) = "0"b;
```

```
          end;
          xt (1) = .5;
/* draw unit contacts for the (jj-1) th pair of adjacent posts */
          do jj = 2 to ihole;
/* calculate x coordinate of post pair in plot inches */
               xt (1) = xt (1)+dxt (jj-1);
               xt (2) = xt (1)+dxt (jj);
/* find the indices of the highest (itop) and lowest (ibase)
   unit for each post */
               do i = 1 to 2,
                    itop (i) = 1;
                    do while
             (post (jj-2+i).unit (itop (i)).unit_present = "0"b);
                         itop (i) = itop (i)+1;
                    end;
                    ibase (i) = 15,
                    do while
            (post (jj-2+i).unit (ibase (i)).unit_present = "0"b);
                         ibase (i) = ibase (i)-1;
                    end;
/* calculate the y coordinate of the highest unit in each hole */
                    yt (i) = scl*
                   (post (jj-2+i).unit (itop (i)).topplt-fmin)+1.;
               end;
               itopblnk = max (itop (1), itop (2));
/* if the top unit is the same for both posts the top contact
   can be drawn */
               if itop (1) = itop (2) then do;
                    call line (xt, yt, 2, 0);
               end;
/* continue drawing contacts until the lowest units have
   been finished */
               do while
              ((itop (1) <= ibase (1))| (itop (2) <= ibase (2)));
/* case for which the unit under consideration is present in
   both holes */
                    if itop (1) = itop (2) then do;
/* ilow indicates for which hole the index of the unit under
   /* consideration for that hole (itop) is lowest (uppermost
   in the sequence of units).  for the case in which the
   top unit is the same for both holes, ilow is arbitrarily
   set to 1 (leftmost hole) */
                         ilow = 1;
/* calculate the y coordinates (yt) of the base of the unit
   being considered (in plot inches) and find the indices
   for the next unit present in each hole */
                         do i = 1 to 2;
                              yt (i) = scl*
                   (post (jj-2+i).unit (itop (i)).basplt-fmin)+1.;
                              itop (i) = itop (i)+1;
                              do while
            ((post (jj-2+i).unit (itop (i)).unit_present = "0"b);
/* stop stepping when the index gets to its max possible value */
```

```
                                        & (itop (i) < 16));
                                    itop (i) = itop (i)+1;
                                end;
                            end;
/* draw the line */
                            call line (xt, yt, 2, 0);
                        end;
/* case for which the unit under consideration is present in only
   one of the holes  */
                        else do;
/* find out for which hole the index of the unit under
   consideration is the lowest(uppermost unit in sequence
    of units) */
                            if itop (1) < itop (2) then ilow = 1;
                            else if itop (1)>itop (2) then ilow = 2;
/* if the hole specified by ilow (1 is left, 2 is right) has
   already reached the lowest unit present in that hole, change
   ilow to indicate the other hole */
                            if itop (ilow) > ibase (ilow) then
                                ilow = 3 - ilow,
/* calculate the y coordinate of the base of the unit under
   consideration in the hole specified by ilow.  the y
   coordinate for the other hole is not changed because
   the unit pinches out at that hole */
                            yt (ilow) = scl*
                (post (jj-2+ilow).unit (itop (ilow)).basplt-fmin)+1.;
/* if the unit is present in both holes or if there is a
   lower unit indicating a pinch out draw the line
   otherwise do not draw */
                            if itop (ilow) <= ibase (3-ilow) |
                                itop (3-ilow)<ibase (3-ilow)
                                then drawok = "1"b;
                            else drawok = "0"b;
/* find the next unit present in the hole specified by ilow */
                            itop (ilow) = itop (ilow)+1;
                            do while ((post (jj-2+ilow).unit
                                (itop (ilow)).unit_present = "0"b)
                                & (itop (ilow) < 16));
                                itop (ilow) = itop (ilow)+1;
                            end;
/* if the unit under consideration in the hole specified
   by ilow has been bounded from above by a unit in the other
   hole, then the contact can be drawn */
                            if (itop (ilow) >= itopblnk & drawok)
                                then call line (xt, yt, 2, 0),
                        end;
                    end;
                    if jj = 2 then do;
/* finish annotation for the first (leftmost) hole */
                        call maklab (post (1), xt (1), fmin, scl,
                        labmem);
                        call holelab (xt (1), post (1).charid, 1);
                        call boreline (post (1), xt (1), fmin, scl);
```

```
                end;
                call naklab (post (jj), xt (2), fmin, scl,
                     labmem);
                call holelab (xt (2), post (jj).charid, jj);
                call boreline (post (jj), xt (2), fmin, scl);
           end;
           call drawyaxf (fmin, fmax, ypage);
/* terminate plot and end */
           call stopit;
           call dsr (">iml>disspla");
     end;
```

****

****

pagin.pll

```
pagin: proc (post);
/* procedure to get page-header information about a hole */
%include postl;
dcl  j fixed bin;
dcl  ans char (1);
dcl (sysin, sysprint)file;
dcl  conversion condition;
           on conversion begin;
/* procedure to handle the input of a character value into a
   numeric variable */
                put skip list ("Please enter a number.");
                go to start;
           end;
           ans = "n";
/* get the information */
start:     do while (ans = "n");
                put skip list ("Insert surface elevation:  ");
                get list (elevation);
                put skip (0) list
                    ("Insert latitude in deg, min, and sec:  ");
                get list ((latitude (j) do j = 1 to 3)),
                put skip (0) list
                     ("Insert longitude in deg,min and sec:  ");
                get list ((longitude (j) do j = 1 to 3));
/* print the information for checking */
                put edit ("id:  ", charid, "elevation:  ",
                     elevation)
                     (a (6), a (8), x (2), a (11), f (12, 2));
                put edit ("latitude", latitude (1), "deg",
```

```
                    latitude (2), "min", latitude (3), "sec")
                    (skip, x (3), a (8), x (5), (2) (f (8, 0),
                    x (1), a (3), x (3)),
                    f (8, 1), x (1), a (3));
               put edit ("longitude", longitude (1), "deg",
                    longitude (2), "min", longitude (3), "sec")
                    (skip, x (3), a (9), x (4), (2) (f (8, 0),
                    x (1), a (3), x (3)),
                    f (8, 1), x (1), a (3));
/* return if information is ok */
               put skip list ("Are these values OK? (y or n): ");
               get list (ans);
               do while (ans ^= "n" & ans ^= "y");
                    put skip (0) list ("Please insert y or n.");
                    get list (ans);
               end;
          end;
     end;
```

****

****

plotinfo.pll

```
plotinfo: proc (ibaud, xpage, ypage);
/* procedure to interrogate the user for the tektronix speed
   and the page dimensions in plot inches */
dcl (sysin, sysprint) file;
dcl  ibaud fixed bin;
dcl  ans char (1);
dcl (xpage, ypage) float bin;
dcl (baud_ok, pgsize_ok) bit (1);
dcl  lab label;
dcl  clear_sysin entry;
dcl (conversion, error) condition;
          on error begin;
               put skip list ("Error. Start over.");
               go to again;
          end;
/* input error trap */
          on conversion begin;
               put skip list ("Input error. Please enter a number.");
               put skip;
               call clear_sysin;
               go to lab;
          end;
```

```
/* get the tektronix speed */
again:     lab = again;
           baud_ok = "0"b;
           do while (^baud_ok);
                baud_ok = "1"b;
                put edit
     ("Insert your baud rate (ch/sec) for the Tektronix 4014:  ")
                     (skip, a (56));
                get list (ibaud);
                if ibaud ^= 30 & ibaud ^= 120
                & ibaud ^= 960 then do;
                     baud_ok = "0"b;
                     put edit
          ("baud rates (ch/sec) supported are 30, 120, and 960.",
"For other rates, insert the highest number below your rate.")
                          (skip, a (51), skip, a (59)),
                end;
           end;
page:      lab = page;
           ans = "n";
/* get the page dimensions */
           do while (ans ^= "y");
                pgsize_ok = "0"b;
                do while (^pgsize_ok);
                     put skip list
                    ("Insert x and y page dimensions in inches.");
                     get list (xpage, ypage);
                     put edit ("xpage = ", xpage, "ypage = ",
                          ypage, "OK? (y or n)")
                          (skip, a (8), f (7, 1), x (3), a (8),
                          f (7, 1), x (3), a (12));
                     get list (ans);
                     if xpage < 3. | ypage < 4. then do;
                          put skip list
                             ("The page dimension is too small.");
                          put skip list
          ("xpage must be greater than 3, ypage greater than 4.");
                     end;
                     else pgsize_ok = "1"b;
                end;
           end;
      end;
```

****

****

pltcde.pll

```
pltcde: proc (char4, char6);
/* procedure to format the unit labels so that the
   first letter will be drawn in upper case using the
   Issco dissplа convention */
dcl  char4 char (4),
     char41 char (1) def char4 pos (1),
     char42 char (1) def char4 pos (2),
     char43 char (1) def char4 pos (3),
     char44 char (1) def char4 pos (4);
dcl  char6 char (6),
     char61 char (1) def char6 pos (1),
     char62 char (1) def char6 pos (2),
     char63 char (1) def char6 pos (3),
     char64 char (1) def char6 pos (4),
     char65 char (1) def char6 pos (5),
     char66 char (1) def char6 pos (6);
          char61 = "(";
          char62 = char41;
          char63 = ")";
          char64 = char42;
          char65 = char43;
          char66 = char44;
     end;
```

****

****

post.incl.pll

```
dcl 01 post (25),
/* include file for defining the 25 posts used during
   the draw routine */
    02 charid char (8),
    02 elevation float bin,
    02 latitude (4) float bin,
    02 longitude (4) float bin,
    02 unit (16),
      03 unit_present bit (1),
      03 untdptp float bin,
      03 untdptb float bin,
      03 topplt float bin,
      03 basplt float bin;
```

```
*************************************************************************
-------------------------------------------------------------------------
*************************************************************************
```

post1.incl.pl1

```
dcl 01 post,
/* include file for defining the data structure for
   one borehole record */
    02 charid char (8),
    02 elevation float bin,
    02 latitude (4) float bin,
    02 longitude (4) float bin,
    02 unit (16),
      03 unit_present bit (1),
      03 untdptp float bin,
      03 untdptb float bin,
      03 topplt float bin,
      03 basplt float bin;
```

```
*************************************************************************
-------------------------------------------------------------------------
*************************************************************************
```

post2.incl.pl1

```
dcl 01 post2,
/* include file for defining the data structure for
   one borehole record */
    02 charid char (8),
    02 elevation float bin,
    02 latitude (4) float bin,
    02 longitude (4) float bin,
    02 unit (16),
      03 unit_present bit (1),
      03 untdptp float bin,
      03 untdptb float bin,
      03 topplt float bin,
      03 basplt float bin;
```

```
*************************************************************************
-------------------------------------------------------------------------
*************************************************************************
```

post31.incl.pll

```
dcl 01 post1,
/* include file containing the first column variable
   for the dp3 subroutine */
    02 charid1 char (8),
    02 elevation1 float bin,
    02 latitude1 (4) float bin,
    02 longitude1 (4) float bin,
    02 unit1 (16),
      03 unit_present1 bit (1),
      03 untdptp1 float bin,
      03 untdptb1 float bin,
      03 topplt1 float bin,
      03 basplt1 float bin;
```

```
*************************************************************************
-------------------------------------------------------------------------
*************************************************************************
```

post32.incl.pll

```
dcl 01 post2,
/* include file containing the second column variable
   for the dp3 subroutine */
    02 charid2 char (8),
    02 elevation2 float bin,
    02 latitude2 (4) float bin,
    02 longitude2 (4) float bin,
    02 unit2 (16),
      03 unit_present2 bit (1),
      03 untdptp2 float bin,
      03 untdptb2 float bin,
      03 topplt2 float bin,
      03 basplt2 float bin;
```

```
*************************************************************************
-------------------------------------------------------------------------
*************************************************************************
```

post33.incl.pll

```
dcl 01 post3,
/* include file containing the third column variable
   for the dp3 subroutine */
    02 charid3 char (8),
    02 elevation3 float bin,
    02 latitude3 (4) float bin,
    02 longitude3 (4) float bin,
    02 unit3 (16),
      03 unit_present3 bit (1),
      03 untdptp3 float bin,
      03 untdptb3 float bin,
      03 topplt3 float bin,
      03 basplt3 float bin;
```

```
*************************************************************************
-------------------------------------------------------------------------
*************************************************************************
```

read_posts.pll

```
read_posts: proc (post, ihole, abort);
/* procedure to control the selection and reading from the data
   base of the holes selected to be fence posts */
dcl (sysin, sysprint, fence_dat) file;
dcl (ihole, i, k) fixed bin;
dcl  ans char (1);
dcl  charidin char (8);
dcl  abort bit (1);
%include post;
dcl  char_id entry (char (8));
dcl  clear_sysin entry;
dcl (error, key) condition;

          abort = "0"b;
/* in case of unexpected error, close files and abort the
   program */
          on error begin;
```

```
                put skip list
                     ("Unexplained error in read_posts.");
                abort = "1"b;
                go to closeout;
           end;
/* initialize hole counter and open file */
/* in case there is no hole with the given id,
   correct the hole count and get another key */
           on key (fence_dat) begin;
                ihole = ihole-1;
                put skip list ("No such hole.  Try again.");
                call clear_sysin;
                go to gtkey;
           end;
           ihole = 0;
           open file (fence_dat) keyed update;
           charidin = "12345678";
gtkey:     do while ("1"b);
                call char_id (charidin);
/* signal to end data entry if key value is "end" */
                if charidin = "end" then go to closeout;
                ihole = ihole+1;
/* read the record and calculate location in decimal degrees */
                read file (fence_dat) key (charidin)
                     into (post (ihole));
/* change deg, min, and sec to decimal degrees */
                post (ihole).latitude (4) =
                     post (ihole).latitude (1)+
                     post (ihole).latitude (2)/60.+
                     post (ihole).latitude (3)/3600.;
                post (ihole).longitude (4) =
                     post (ihole).longitude (1)+
                     post (ihole).longitude (2)/60.+
                     post (ihole).longitude (3)/3600.;
/* maximum number of holes is 25 */
                if ihole = 25 then go to closeout;
           end;
closeout: close file (fence_dat);
     end;
```

****

****

```
remove_hole: proc;
/* procedure for deleteing records from the fence_dat file
   of borehole information */
dcl  question char (60) varying,
     answer char (1),
     charidin char (6);
dcl (exist_code, delete_loop_continue) bit (1);
%include postl;
dcl  name char (32) init ("fence_dat");
dcl  exist entry options (variable);
dcl  yesno entry (char (60) varying, char (1));
dcl  char_id entry (char (6));
dcl  show_fence_dat entry options (variable);
dcl (sysin, sysprint, fence_dat) file;
dcl  key condition;
/* if no record matches the input key, start over */
          on key (fence_dat) begin;
               put skip list ("No such hole.  Try again.");
               go to gtkey;
          end;
/* check to see whether the data file fence_dat exists */
          call exist (name, exist_code);
/* if no data file exists print an error message and return */
          if ^exist_code then do;
               put edit ("The data file doesn't exist.",
      "You must have a data file before you can delete records.")
                    (skip, a (20), skip, a (58));
               return;
          end;
/* open data file */
          open file (fence_dat) keyed sequential update;
          delete_loop_continue = "1"b;
/* deletion loop, continues until the continue flag is reset */
gtkey:    do while (delete_loop_continue);
               call yesno ("Another hole? (y or n):  ", answer),
/* set continue flag to "0" b if no more holes */
               if answer = "n" then delete_loop_continue = "0"b;
               else do;
/* get the id for the hole to be deleted */
                    call char_id (charidin);
/* if the returned id is not "end" then proceed with deletion */
                    if charidin ^= "end" then do;
/* if user wants to see hole before deletion, read and display
   hole info */
                         call yesno
    ("Would you like to review this hole? (y or n):  ", answer);
                         if answer = "y" then do;
                              read file (fence_dat) key
```

```
                                        (charidin) into (post);
                                   call show_fence_dat (post);
                              end;
/* final check with user before deletion */
                              call yesno
          ("Do you want to delete this hole? (y or n):  ", answer);
                              if answer = "y" then delete file
                                   (fence_dat) key (charidin);
                         end;
/* if input id is "end" set continue flag to "0"b */
                         else delete_loop_continue = "0"b;
                    end;
               end;
/* close file and return */
               close file (fence_dat);
               return;
     end;
```

****

****

select_ref.pll

```
select_ref: proc (refans);
/* procedure to have the user select the method for
   calculating distance from a chosen level datum */
dcl (sysin, sysprint) file;
dcl  refans char (1);
dcl  clear_sysin entry options (variable);
          refans = "0";
/* ask for input */
select:   put edit ("The reference datum can be:",
          "1)  elevation",
          "2)  the top of a unit present in all selected holes",
          "3)  the base of a unit present in all selected holes")
          (skip, a (27), skip, x (5), a (13), skip, x (5),
          a (55), skip, x (5), a (56));
/* check that input is acceptable */
          do while (refans ^= "1" & refans ^= "2" & refans ^= "3"
                    & refans ^= "4");
               put edit ("Insert 1, 2, 3, or 4 to exit:  ")
                    (skip (2), a (31));
               get list (refans);
               call clear_sysin;
          end;
     end;
```

```
*****************************************************************
-----------------------------------------------------------------
*****************************************************************
```

show_fence_dat.pll

```
show_fence_dat: proc (post);
/* procedure to print the record for one hole */
dcl (sysin, sysprint)file;
dcl  i fixed bin;
%include postl;
dcl  unit_code (16) char (4) external static;
/* print the header information */
          put edit ((40)"*") (skip, a (40));
          put edit ("id:  ", charid, "elevation:  ", elevation)
               (skip, a (6), a (8), x (2), a (11), f (12, 2));
          put edit ("deg", "min", "sec")
               (skip, x (14), a (3), (2) (x (5), a (3)));
          put edit ("latitude", latitude (1), latitude (2),
               latitude (3))
               (skip, a (8), x (3), (2) (f (6, 0), x (2)),
               f (6, 1));
          put edit ("longitude", longitude (1), longitude (2),
               longitude (3))
               (skip, a (9), x (2), (2) (f (6, 0), x (2)),
               f (6, 1));
/* print the unit data only for those units present in the
   hole */
          put edit ("top depth", "base depth")
               (skip, x (12), a (9), x (3), a (10));
          do i = 1 to 15;
               if unit (i).unit_present = "1"b then do;
                    put edit (unit_code (i), unit (i).untdptp,
                         unit (i).untdptb)
                         (skip, a (4), x (6), f (9, 1), x (3),
                              f (9, 1));
               end;
          end;
          put edit ((40)"*") (skip, a (40));
          put skip;
     end;
```

```
*****************************************************************
-----------------------------------------------------------------
*****************************************************************
```

stopit.pll

```
stopit: proc;
/*  procedure to terminate the plot */
dcl  endpl entry options (variable);
          call endpl (0);
          return;
     end;
```

****

****

strtmap.pll

```
strtmap: proc (ibaud, xpage, ypage);
/* procedure to initialize the tektronix and plot the
   axes and title information */
dcl (ibaud, izero, hund) fixed bin;
dcl (xpage, ypage, xpltpg, ypltpg, zero, one, fhite, tlen,
     xtit, ytit, xor, yor) float bin;
dcl  titlin char (100) varying;
dcl  blank char (1);
dcl (tk30, tk120, tk960, setup_tektronix_tcs) entry;
dcl (page, title, graph, basalf, mixalf, height, line,
     physor, messag)
     entry options (variable);
dcl  xmess entry options (variable) returns (float);
dcl (sysin, sysprint)file;
/* initialize tektronix terminal control system */
          put edit ("Insert title (<32 char) followed by $.  ")
               (skip, a (40));
          put edit
            ("Multiple word titles must be enclosed in quotes: ")
               (skip, a (50));
          get list (titlin);
          fhite = .2;
          call setup_tektronix_tcs;
/* inform disspla of baud rate */
          if ibaud = 30 then call tk30;
          else if ibaud = 120 then call tk120;
          else if ibaud = 960 then call tk960;
/* call disspla routines setting page size, title
   annotation, and axis scaling */
```

```
            call page (xpage, ypage);
            xpltpg = xpage-2.;
            ypltpg = ypage-3.;
            call height (fhite);
            hund = 100;
            izero = 0;
            blank = " ";
            zero = 0.;
            one = 1.;
            xor = 1.2;
            yor = 1.;
            call physor (xor, yor);
            call title (blank, izero, blank, izero, blank, izero,
                 xpltpg, ypltpg);
            call graph (zero, one, zero, one);
            tlen = xmess (titlin, hund);
            xtit = (xpage-tlen-2.)/2.;
            ytit = ypage-1.75;
            call messag (titlin, hund, xtit, ytit);
/* call disspla routines to set character type */
            call basalf ("1/cstd");
            call mixalf ("standard");
            return;
      end;
```

****

****

unit.pll

```
unit: proc (post, n),
/* procedure to get the unit data for the n-th unit in a hole
   as specified by input of n */
dcl (sysin, sysprint)file;
dcl  n fixed bin;
dcl  tmpin float dec,
dcl  unit_code (16) char (4) external static;
dcl  tmpchr char (20) varying,
dcl  question char (60) varying;
dcl  answer char (1);
dcl  clear_sysin entry options (variable);
dcl  yesno entry (char (60) varying, char (1));
/* condition to check for input of a character value into a
   numeric variable */
dcl  conversion condition;
dcl (onchar, convert, fixed) builtin,
```

```
dcl  ans char (1);
%include postl;
          on conversion begin;
               put skip list
                  ("input error at: ", onchar, " :start again");
               call clear_sysin;
               put skip;
               go to start;
          end;
/* read depth to unit top and verify entry */
          answer = "n";
start:    do while (answer = "n");
               put edit ("Insert unit depth to top", ": ")
                   (skip (0), a (24), x (15), a (2));
               get list (tmpin);
               call clear_sysin;
/* construct verification question */
               question = unit_code (n)||"  "||convert (tmpchr,
                    fixed (tmpin, 10, 2))||"    OK? (y or n):  ";
               call yesno (question, answer);
               unit (n).untdptp = convert (unit (n).untdptp,
                    tmpin);
          end;
     end;
```

****

****

unit_control.pll

```
unit_control: proc (post, abort);
/* procedure to control the input of unit data for all units
   of one hole */
dcl (sysin, sysprint) file;
dcl (clear_sysin, unit, decode_unit) entry options (variable);
dcl  i fixed bin;
dcl  ans char (1);
dcl  unit_code (16) char (4) external static;
%include postl;
dcl  code char (4);
dcl (unit_p, unit_n)fixed bin;
dcl (abort, top_ok, mid_ok) bit (1);
          abort = "0"b,
/* get code of top unit in character form */
          top_ok = "0"b;
          do while (^top_ok),
```

```
                put edit ("Insert top unit code", ": ")
                     (skip, a (20), x (19), a (2), a (4));
                get list (code);
                call clear_sysin;
/* decode the character unit code, check that the code is
   acceptable, and return the integer index for that code */
                call decode_unit (code, unit_p);
                do while (unit_p = 0);
                     put skip list ("Hole voided.");
                     abort = "1"b;
                     return;
                end;
/* check for reasonable value for the unit index number */
                if unit_p >= 17 then begin;
                     top_ok = "0"b;
                     put skip list
                  ("Top unit can't be the bottom of the hole.");
                end;
                else top_ok = "1"b;
/* get the information for the top unit in the hole */
           end;
           call unit (post, unit_p);
           unit (unit_p).unit_present = "1"b;
/* get all units present between the top unit and the bottom
   unit in the hole */
           mid_ok = "0"b;
           do while (^mid_ok);
                put edit ("Insert next lower unit code", ": ")
                     (skip, a (27), x (12), a (2));
/* get the unit code and find the corresponding index */
                get list (code);
                call clear_sysin;
                call decode_unit (code, unit_n);
/* check for reasonable unit index */
                do while (unit_n = 0);
                     put skip list ("Hole voided.");
                     abort = "1"b;
                     return;
                end;
/* if new unit is not below the previously entered unit, there
   has been an error.  try again for the next unit. */
                if unit_n <= unit_p then do;
                     put skip list
                     ("New unit code not below than last one."),
                     do i = 1 to 17;
                          if unit (i).unit_present = "1"b then
                               unit_p = i;
                     end;
                     put edit
          ("Last valid unit entered was", ": ", unit_code (unit_p))
                          (skip, a (27), x (12), a (2), a (4));
                end;
                else if unit_n <= 15 then do;
```

```
                    call unit (post, unit_n);
/* check that the new unit is deeper than the last one */
                    if post.unit (unit_n).untdptp <=
                         post.unit (unit_p).untdptp then do,
                         put skip list
               ("The new unit is not deeper than the last one."),
                         put skip list
                         unit_n = unit_p;
                    end,
/* set the bottom of the unit above to the top value of the
   unit just inserted */
                    unit (unit_n).unit_present = "1"b;
                    unit (unit_p).untdptb =
                         unit (unit_n).untdptp;
                    unit_p = unit_n;
               end;
               else mid_ok = "1"b;
/* get base of the lowest unit present */
          end;
          put edit ("Insert base of lowest unit measured", ": ")
               (skip (0), a (35), x (4), a (2)),
          get list (unit (unit_p).untdptb);
          return;
/* error return with abort flag set */
erret:    put edit ("hole voided") (skip, a (11));
          abort = "1"b;
          return;
     end;
```

****

****

yesno.pll

```
yesno: proc (question, answer);
/* utility program to print the input question
   and keep prompting until a y or n answer has
   been input */
dcl  question char (60) varying;
dcl  answer char (1);
dcl (sysin, sysprint) file;
dcl  okyesno bit (1);
dcl  clear_sysin entry;
          okyesno = "0"b;
          do while (^okyesno);
               put skip list (question);
```

```
            get list (answer);
            call clear_sysin;
            if answer = "y" | answer = "n" then
                 okyesno = "1"b;
            else put skip list ("Please answer y or n."),
        end;
    end;
```

****

****