

init:

```
*=====
| The following pmenu call applies to the
| primary (program) window.
*=====;
rc = pmenu('afib.sasview.edit.pmenu');

* TURN PMENUs ON ...;
call execcmdi('pmenu');

dsid = open('maps.afghanis','i');

cval = "";

* SIZE THE SELECTION LIST WINDOW ...;
call wregion(4,20,15,40);
call execcmd('setpmenu afib.sasview.abort.pmenu');

* DISPLAY SELECTION LIST ...;
cval = varlist(dsid,'all',1);

rc = close(dsid);

return;
```

Exhibit 1

It is necessary to delay the invocation of the SETPMENU command until control is returned to the application after the selection list has been displayed. SCL function EXECCMD, then, must be used instead of EXECCMDI, which would execute the SETPMENU command immediately and incorrectly associate the pmenu ABORT with the primary display window instead of the selection list window. This appears to work flawlessly, until the function VARLIST is replaced by DIRLIST.

Problems

The same code now causes the pmenu ABORT to be associated with the primary program display window instead of the selection list window that is generated by DIRLIST. CATLIST -- and probably some other selection list functions as well -- exhibits the same behavior.

There are also other anomalies. The problem mentioned above is resolved if the pmenu call is placed in MAIN: rather than INIT:. Sometimes using the SCL function REFRESH solves pmenu association problems. On a separate but related note, if the program entry is an extended table, the program entry's primary display

window does not resize when the selection list window has closed.

The problem revolves around the SCL functions that comprise the 'pmenu association' tool set:

EXECCMD: *executes it's argument when control returns to application*

EXECCMDI: *executes it's argument immediately*

PMENU: *displays pmenu named in it's argument when procedure redisplay a window*

SETPMENU: *assigns pmenu entry to the current window (documented in FSP reference under PMENUNAME)*

It is a non-trivial task to determine when 'control is returned to the application' or 'the procedure redraws the window.' A simple construction, however, makes a developed understanding of these details unnecessary.

Methodology Revisited

These problems are all resolved when the selection list function is called in a separate catalog entry of type SCL that is invoked via a call to DISPLAY. The new code appears in exhibit 2.

init:

```
*=====
| The following pmenu call should apply to the display
| window.
*=====;
rc = pmenu('afib.sasview.edit.pmenu');
call execcmdi('pmenu');

dsid = open('maps.afghanis','i');

cval = "";
call wregion(4,20,15,40);
call execcmd('setpmenu afib.sasview.abort.pmenu');

call display('dirlist.scl','maps',cval);

rc = close(dsid);

return;
```

Exhibit 2

The entry DIRLIST.SCL appears in exhibit 3.

```

=====
Displays dirlist (dataset list) window with custom
pmenu.

parameters:
  libname: libref for directory to display
  memname: character variable that will contain
           chosen member name
=====;

init:

entry libname $8 memname $20;

call wregion(5,14,15,54,"");

* NOTE 4-LEVEL PMENU NAME! ;
call execcmd('setpmenu afill.sasview.abort.pmenu');
memname =
  dirlist(libname,'data',1,'Y','','',
  'Choose a dataset (position cursor, press <Return>');

return;

```

Exhibit 3

The only drawback to this technique is that a separate SCL entry (like DIRLIST.SCL) is required for each selection list function that will be used in the program entry or application. This can be circumvented, though, by creating an SCL entry that contains a series of method blocks (new to version 6.07), one for each selection list function that will be used in an application. Such an entry is illustrated in exhibit 4.

```

DIRLIST:
method libname $ memname $ ;

call wregion(5,14,15,54,"");
call execcmd('setpmenu afill.sasview.abort.pmenu');
memname = dirlist(libname,'data',1,'Y','','',
  'Choose a dataset (move cursor, press <Return>');
endmethod;

VARLIST:
method dsid 8 varname $ ;

call wregion(5,14,15,54,"");
call execcmd('setpmenu afill.sasview.abort.pmenu');
varname = varlist(dsid,'all',1);
endmethod;

```

Exhibit 4

If the SCL entry in exhibit 4 was named SELLIST.SCL, then the VARLIST method within that entry could be called using:

```
call method('sellist.scl','varlist',dsid,cval);
```

SELLIST.SCL could be further modified so that the methods defined within it could take window sizes (used in the WREGION function) or custom pmenu names as arguments.

It is possible, incidentally, to hide the command line and suppress pmenus altogether. To this end, the code fragment in exhibit 5 generates a null pmenu.

```

proc pmenu;

menu null;
  item "";
run;

```

Exhibit 5

Use the technique illustrated earlier to associate this custom pmenu with the selection list generated by the varlist function and obtain:

```

+-----+
+Variables-----+
+-----+
Make one selection.
Data set:  MAP5.ASIA                      Copyright (C) 1989
Name      Type  Len  Label
ID         NUM   3    Country ID Number
SEGMENT    NUM   3    Country SEGMENT Number
X          NUM   4    Projected Longitude
Y          NUM   4    Projected Latitude

```

The symbol  cannot be suppressed.

Conclusions

This poster is essentially a response to a defect in pull-down menu implementation in the current version of SCL. The technique illustrated in exhibits 2, 3, and 4 is slightly circumspect but reliable and obviates the need for advanced understanding of some of the subtleties of SCL.

Author Contact Information

Comments and suggestions are welcome. Please contact:

Paul Dalberth
Family Health International
Durham, NC 27713
(919) 544-7040 ext. 535