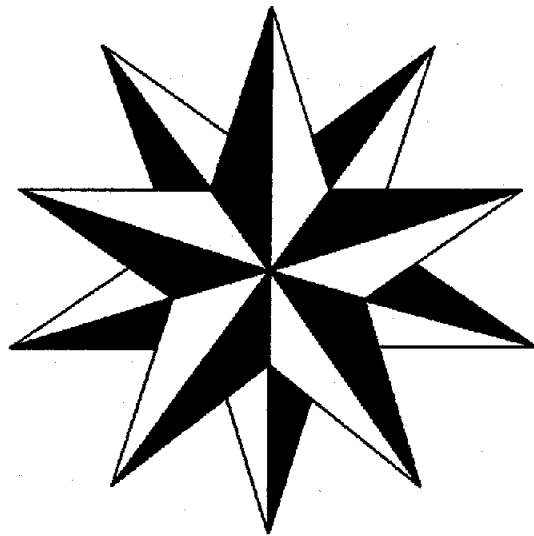


**SCREEN CONTROL LANGUAGE - AN
INVALUABLE TOOL IN APPLICATIONS DEVELOPMENT**



**David Septoff and Don Henderson
SAS Consulting Services, Inc.**

Screen Control Language - An Invaluable Tool in Applications Development

David S. Septoff, SAS Consulting Services Inc., Rockville, MD

ABSTRACT

Screen Control Language (SCL) is a programming language that provides functions and routines to control fields on windows and screens to manage the windows and screens that make up an application. With SCL, the SAS* programmer can

- perform real-time cross-validation of variables
- control the application and interact with the SAS System
- control access to, and obtain information from, SAS data libraries
- control communication in an application.
- control error handling and provide messages specific to the application

This tutorial demonstrates that SCL provides the power and flexibility needed to build all types of applications, from simple programs that accomplish a few tasks to sophisticated SAS systems.

INTRODUCTION

The purpose of this tutorial is to introduce the user to SCL as an applications development tool. A short definition of SCL and its uses will begin the tutorial, followed by a discussion on the issues confronting the applications developer and how SCL can address these issues, and finally the tutorial will draw some conclusions on using SCL to facilitate applications development. Note: Throughout the tutorial certain terms are used which are characteristic of SCL. This tutorial is task oriented; therefore it does not spend time defining terms that can be found in the documentation. Please see the references at the end of the tutorial for assistance in the definition of terms.

SCL, as defined in *SAS Screen Control Language: Reference, Version 6, First Edition*, is a language designed to assist in the development of interactive applications using the SAS System. SCL provides the power that a programmer needs to build all types of applications. SCL combines the syntax of the SAS base language with other statements and functions that enable the applications developer to control the flow of the application based on input from the user and/or the data within the application.

SCL facilitates the development of data-driven applications. Data-driven applications are developed so that data drive the system. Parameter files are used to allow the system administrator to control the use of the system. Parameter files can

- list the valid users of the system
- list the valid selections for a variety of control variables (for example, user types, organization codes, and so on)
- list available reports
- list parameter files.

This gives the system administrator the ability to add reports and parameter files to the system without any changes to the basic code of the system. To add a report, the system administrator would add the report name to the correct parameter file. To add a new parameter file, the system administrator would add the name of the new

file to the list of parameter files. The system will automatically add the new file to the parameter file maintenance window. This structure ensures flexibility and ease of maintenance.

MENUS

In any interactive application, menus are used as the mechanism to access the functions of the system. SCL, used along with SAS/AF* software, allows for several types of menus. This section illustrates some of those menu types and explains the positive and negative aspects of them.

MENU Entries

Through MENU entries within SAS/AF software, the programmer may build menus that provide lists of selections to users. Chosen selections are accessed through additional MENU entries, PROGRAM entries and so on that navigate the user through the application. MENU entries require that no SCL be written in order to execute the user's choice; a select option is used. MENU entries however do require that the user provide a selection value.

PROGRAM Entries

PROGRAM entries are an extremely powerful and versatile tool for building menus. PROGRAM entries do not require the user to actually enter a value. They require that the user tab to the selected option and press ENTER. PROGRAM entries do require SCL code to direct the execution of the user's choice. There are several ways that PROGRAM entries can be used as a menu, depending on the functions used in SCL and the PROGRAM entries' attributes.

BLOCK function

The BLOCK function displays a menu (similar to one in SAS/ASSIST* software) containing choice blocks with labels of the functional pieces of the application. The user tabs to the appropriate selection and presses the ENTER key. The PROGRAM entries associated with the SCL program execute the user's selection. The following SCL code executes a block type menu:

```
INIT;  
RETURN;  
  
MAIN;  
choice = 2; /* places cursor on selection #2 on window */  
do while (choice ne 0) ;  
    choice = BLOCK('Sample Executive Information System',  
                  'Welcome to the Sample SIS',  
                  'Global Params', 'Reports',  
                  'Utilities', 'End',  
                  ' ', ' ', ' ', ' ', ' ',  
                  ' ', ' ', ' ', ' ', ' ');  
    select(choice);  
    when(0) return;  
    when(1) call display('globals.program');  
    when(2) call display('reptmenu.program');  
    when(5) call display('utility.program');  
    when(6) call display('endsas.program');  
    otherwise return;  
end;  
end;  
RETURN;  
  
TERM;  
RETURN;
```

For this PROGRAM entry the BLOCK function is executed within a loop. There are no window variables, and therefore the INIT,

MAIN, and TERM sections execute in sequence. The number of block labels specified determines the number of blocks displayed on the menu. Depending on the display device, the function gives the applications developer 31 different color schemes for displaying the blocks. The advantage of this type menu is its user friendliness. The display is presented in icon-like manner, which is visually satisfying for someone not comfortable with computers. The disadvantages are that it does not allow for entry on the screen. Therefore, performing what should be a very simple task might take several menus to complete.

Choice Groups

For a PROGRAM entry, the applications developer can design a group of fields that display values from which users can make a selection by placing the cursor on the selected item and pressing ENTER. Fields that belong to a choice group are mutually exclusive, which means that users can only select one item. See *SAS/AF Software: Usage and Reference, Version 6, First Edition* for help with the correct attributes for setting up choice groups. Following is sample SCL code to execute a choice group type menu screen with three selections:

```
INIT:
RETURN;

MAIN:
select(chcgrp); /*chcgrp is the Choice Group Name*/
when('1') call display('prntto.program');
/* the following command is operating system dependent */
when('2') call execcmd('x prntoff sasds.prtfile;');
when('3') call display('parmnu.program');
end;
RETURN;

TERM:
RETURN;
```

Much like the block-type menu, the choice group only allows for one selection at a time. But the fact that the selections are being presented in groups allows for more flexibility for the developer. For example, two groups of entries may appear within the window. The first group, action, has two selections: edit and print. The second group could be a list of data sets to be edited or printed. The user can only select one action with one data set.

Extended Tables

The previous menu types that have been referenced have demonstrated fixed menus (the menus do not change dynamically). An extended table is a special type of PROGRAM entry that allows the developer to repeat a set of fields defined in the window. SCL can then be used to process these fields as a table rather than as one row. Each field can be an observation in a SAS data set that is being accessed by the SCL code. This brings us back to the discussion regarding the data-driven nature of applications development, because an extended table can be a window displaying the reports parameter file of all valid reports of the system. Other functions are available for verifying the existence of the report code but for now our discussion will focus on the SCL code for the extended table itself. The following code demonstrates listing the report parameter file in an extended table:

```
INIT:
dsid = open('parm.reports');
call setrow(attrn(dsid,'nobs'),1);
RETURN;

MAIN:
RETURN;

GETROW:
rc = fetchobs(dsid,_currow_);
rname = getvarc(dsid,varnum(dsid,'reptname'));
RETURN;

PUTROW:
```

```
if issel(_currow_) then
call display(rname||'.program');
RETURN;

TERM:
dsid = close(dsid);
RETURN;
```

The user once again only has to tab to the desired selection and press ENTER. Once the selection is made, the system will detect which observation is being accessed; it will then use the value of RNAME from the selected row on the extended table. The code then performs a call display on the PROGRAM entry corresponding to the value of RNAME. The advantage of using the extended table is that the developer builds in the flexibility of allowing users to add functionality by updating data sets without having to perform additional development.

CONTROLLING ACCESS

In applications development, another function that is generally required is controlling access to the system and/or to functions within the system. SCL provides two ways to do this. Both methods assume the existence of a file of valid users and their level of access.

Control at the System Level

You cannot always assume that the operating system protects the system from users who should not be using an application. Therefore, it becomes necessary to control access at a system level or at time of entry into the system. SCL facilitates this by allowing the system to look at a file before it returns controls to the user and a user with an invalid userid is refused entry. The following code demonstrates system level control:

```
INIT:
userid = symget('sysjobid');
dsid = open('parm.userlist');
rc = locatec(user,varnum(user,'userid'),userid);
if rc = 0 then /* userid not found */
call display('endsas.program');
rc = close(dsid);
RETURN;

MAIN:
other SCL statements ....
RETURN;

TERM:
RETURN;
```

In the above example, the application reads the userid from the system macro variable SYSJOBID using the SYMGET function. The parameter file that contains all userids is then opened to determine if the userid exists in the data set (using the LOCATEC function). If the userid is not found, then the user is logged out of the application before ever having control.

Control at the Function Level

There are times when a developer would like to build an application where functions within the system are accessed based on the user type defined in the parameter file of userids. SCL allows the developer to Grey Bar fields on a window using the ACTIVATE function. The ACTIVATE function activates the specified field of a choice group. An example of how to use the ACTIVATE function to Grey Bar a choice follows:

```
INIT:
userid = symget('sysjobid');
dsid = open('parm.userlist');
rc = locatec(user,varnum(user,'userid'),userid);
if rc = 0 then /* userid not found */
call display('endsas.program');
else
if getvarc(user,varnum(user,'admin'))='N' then
```

```

/* grey bar access to choice 3 */
rc = activate('chcgrp',-3);
rc = close(ds1d);
RETURN;

MAIN:
  other SCL statements ....
RETURN;

TERM:
RETURN;

```

In the example above, the application tests to see if the user is allowed access to the administrative functions of the system. If the value 'N' is returned, the application Gray Bars choice group option 3, thus preventing the user from selecting it.

Another way to control access is accomplished by incorporating extended tables. The system could subset the data set that is being used for the menu based on the user's access level. Consequently, when the table is built, only the menu options that are appropriate for that user's access level appear. The code below demonstrates this capability.

```

length userid $8;

INIT:
  userid = symget('sysjobid');
  user = open('S.USERLIST');
  rc=locatec(user,varnum(user,'USERID'),userid);
  if getvarc(user,varnum(user,'ADMIN')) = 'N' then
  do;
    submit continue;
    data reports;
      set s.reports(where=(access ne 'ADMIN'));
    run;
  endsubmit;
  dsid2 = open('work.reports');
  end;
  else dsid2 = open('s.reports');
  call setrow(attrn(dsid2,'nobs'),1);
RETURN;

MAIN:
RETURN;

GETROW:
  rc = fetchobs(dsid2,_currow_);
  reptname = getvarc(dsid2,varnum(dsid2,'reptname'));
RETURN;

PUTROW:
  if issel(_currow_) then do;
    rc = fetchobs(dsid2,_currow_);
    rname = getvarc(dsid2,varnum(dsid2,'rname'));
    call display(rname,'.program');
  end;
RETURN;

TERM:
RETURN;

```

In the INIT section above, the userid is compared against the user-list parameter file. The GETVARC function is then used to retrieve the access level of the user. The system then subsets the report data set, using the WHERE data set option, for reports that this user has access to. The table is then created with only the reports that the user is allowed to see.

DATA ENTRY AND VALIDATION

In the development of software applications there is almost always a data entry mechanism. The data entry mechanism used in developing applications in the SAS System is SAS/FSP® software. SCL can be used as the programming language for SAS/FSP screens. An important part of any data entry mechanism is the real-time validation of the data. Also, equally as important, is ensuring that the data entry application has the capability to assist the user in data

entry (for example, lists of valid values). SCL provides several functions that facilitate this process.

Note: The examples below demonstrate functions for character variables. Corresponding functions exist for numeric variables. Those functions have an N as the last letter of the function name:

- LOCATEC** searches a SAS data set for an observation that contains a specified character value
- DATALISTC** displays a selection list containing observations from a data set for selected variables. The user can position the cursor on a row and press ENTER to select an observation. The value of the specified variable is returned
- SHOWLIST** opens a developer-defined selection list. The user can position the cursor on a row and press ENTER to select a value. The value of the selection is returned.

An example of the use these functions follows:

```

MAIN:
  if scvar1 = _blank_ or modified(scvar1) then
  do; /* validate/pop-up selection list for scvar1 */
    found = LOCATEC(dsid,dvar1,scvar1);
    /* dsid = SAS data set reference */
    /* dvar = data set var reference */
    /* scvar1 = screen var reference */
    if not found then
    do; /* present list */
      call wregion(7,3,17,60);
      /* defines next window */
      scvar1=DATALISTC(dsid,'d1_var1 d1_var2',
        'Please select a value');
    end; /* present list */
  end; /* validate/pop-up selection list for scvar1 */
  if scvar1 = 1 then
    team = showlist('Mets','Cubs','Cards',
      'Select your team');
  else
    if scvar2 = 2 then
      team = showlist('Yankees','Orioles',
        'Select your team');
RETURN;

```

The main difference between the DATALISTC function and the SHOWLIST function is that with the DATALISTC function a data set is being displayed and can be altered by changing the data, where as with the SHOWLIST function the list is hardcoded. The above code illustrates only one application of these functions as well as only listing three of the many functions that are available to the developer.

TOOLS

Within SCL as within base SAS software, there are functions and routines that are consistently used throughout the development process and across applications. The examples presented in this tutorial could be used as tools across applications (for example, the examples for the LOCATEC and DATALISTC functions). This enables the user to create a library or toolbox of SCL routines that are used consistently across applications, much like a library of sub-routines. The developer would create a catalog of SCL toolbox routines that can be accessed when needed in the development process by simply using a CALL DISPLAY or a GOTO function. One routine that is used in the development of menu-driven interactive applications is a confirm/cancel routine. In the development of menu-driven applications developers consistently use a routine such as this to confirm an entry from the user. An example of this routine follows:

```

entry response $1;
INIT:
RETURN;

MAIN:
    response = confcan;
    _status_ = 'H';
RETURN;

TERM:
    msg = 'Select One Choice';
    _status_ = 'R';
RETURN;

```

This routine is called from another window. The value created in this routine is passed back to the calling window through the ENTRY parameter. This routine further illustrates the examples shown earlier in the paper, such as choice groups (the variable CONFCAN is a choice group). This demonstrates that the development of modular components to a system is possible in SCL as well as in base SAS programming.

CALLS OTHER FULL SCREENS

SCL allows the developer to open windows, real-time, of different SAS products based on the value entered (a good example of this is shown in the *SAS Screen Control Language: Reference, Version 6, First Edition*). For example, from within an FSVIEW screen a SAS/AF window can be opened if a value is out of range. Another example might be that if a developer wanted to have the user confirm a request such as when the user was leaving an application. A SAS/AF window could open when the user tries to exit, containing the toolbox routine described earlier - CONFCAN to confirm the request.

CONCLUSION

The purpose of this tutorial is to introduce the user to SCL as an applications development tool. It shows that the use of SCL can sat-

isfy many of the needs of the applications developer. Through the use of SCL, the developer can interact with the user much easier and much better than ever before. Therefore, the developer can create applications that satisfy the users. That is what makes Screen Control Language an invaluable tool in applications development.

ACKNOWLEDGEMENTS

The author wishes to thank Don Henderson and Philip Shelton for their help in reviewing this paper.

The author may be contacted at

SAS Consulting Services Inc.
1700 Rockville Pike
Suite 330
Rockville, MD 20852
(301) 881-8840

REFERENCES

SAS/AF Software: Usage and Reference, Version 6, First Edition
SAS/FSP Software: Usage and Reference, Version 6, First Edition
SAS Screen Control Language: Reference, Version 6, First Edition

SAS, SAS/AF, SAS/ASSIST, and SAS/FSP are registered trademarks of SAS Institute Inc., Cary, NC, USA.