USING SAS/AF, SAS/FSP, AND SCL
TO CREATE A USER-FRIENDLY DATA ENTRY SYSTEM

Elizabeth Li, Kaiser Permanente Division of Research, Oakland, CA.

ABSTRACT

SAS/AF, SAS/FSP, and Screen Control Language (SCL) allow you to create a user-friendly data entry system. This paper presents an example of a system that uses SAS/AF and SAS/FSP to interface with users. The system contains screens for data entry, data retrieval, and report generation. Data safety and validation techniques also are addressed.

The paper presents techniques to build a data entry system with SAS/AF and SAS/FSP. The approach can be applied to data entry in any field. Although, the system is built on PC-SAS, these techniques can be used on any platform.

INTRODUCTION

In data collection for cardiac studies, a large number of 12-lead Electrocardiographs (ECGs) have to be analyzed. The user-friendly system, Automated ECG Coding System, allows cardiologists to record the results of the ECG analysis directly into computers, bypassing paper forms and data entry personnel. A study coordinator can access the system, generate study progress reports and view the data. An independent cardiologist can verify the results analyzed by two other cardiologists respectively. Features of the system include validation of data values when they are entered, editing and browsing of data sets, and generating reports. The system has full-screen interaction between users and application. This paper illustrates how PROC BUILD, PROC FSEDIT, SCL and SAS DATA steps can be used to create these features in system development.

PASSWORD PROTECTION

The main entry screen of the system serves as a security check. Based on the password entered by a user, the system determines which functions or screens that can be accessed by the user. A menu with selections or data entry screens will be available to the user after his or her password is entered. Starting with SAS/AF PROC BUILD, you can create a main entry screen (Figure 1). The build procedure can be used as following:

```
libname ecg '\ecg\basessd';
libname library '\ecg\basessd';
proc build catalog=ecg.main;
run;
```

Figure 1. Starting proc build

To create a password protection program, for example, password.program, you can type at the command line of BUILD: DIRECTORY ECG.MAIN (E) screen (Figure 1):

```
edit password.program
```

Figure 2 Editing password.program entry screen

and create a main entry screen (Figure 2). To give a user data entry screens, you can use call fseedit(). To give a user a menu with several choices, you can use call display() to either call for another program or another menu. The SCL program under BUILD: SOURCE PASSWORD.PROGRAM (Figure 3) is presented here:
init:
control always;
cursor password;
return;
main:
erroroff all;
if password = 'USER1' then do;
password= blank;
call fseedit ('ecg.basicecg','ecg.basicecg','edit');
end;
else if password = 'USER2' then do;
password= blank;
call display('coordin.program');
end;
else if password = 'USER3' then do;
password= blank;
call display('viewmenu.menu');
end;
else do;
erroron password;
alarm;
cursor password;
mag='
Unauthorized Password !!!';
password= blank;
cursor password;
end;
if status = 'E' or _status_ = 'C' then do;
call execcmd('endsas');
end;
return;
term:
return;

Figure 3. Editing password.program SCL

There are three sections in SCL, init, main, and term. In the init section, the control option is to override the default flow of execution. The cursor statement controls the initial cursor position when the password.program is initiated. Erroroff statement clears the error flags. Whereas Erroron statement sets the error flags. _STATUS_ is a system variable.

When _status_ equals 'E', it indicates an 'END' command was issued. When _status_ equals 'C', a 'Cancel' command was issued. Many call routines can be called from SCL. FSEDIT call routine invokes FSEEDIT procedure and passes all control to the procedure. A call FSEDIT routine with specification of 'browse' instead of 'edit' invokes FSBROWSE procedure. EXECCMD call routine executes one or more SAS commands. DISPLAY calls for another menu, program, help, or CBT created by SAS/AF.

CREATING DATA ENTRY SCREENS

The creation of data entry screens can be divided into three basic steps: creation of SAS data sets, customizing data entry screens and setting basic value checks for the data sets with PROC FSEDIT, and programming data validation with SCL. Automatic data filling that increases data entry efficiency and reduces data entry error can be built by the SCL program also.

Creating a SAS data set

One way to create a SAS data set is to use SAS/FSP PROC FSEDIT, and specify new option or use SAS DATA steps. SAS DATA step is more convenient because it is easier to revise the data set and to document. You can revise your data steps for adding, deleting, and changing variables. A sample program called basicecg.sas has the following codes.

```
libname library '\ecg\base.sd';
libname ecg '\ecg\base.sd';
%let title=PATIENT ECG INFORMATION;
data ecg.basicecg (label='Patients ECG information');
attrib timentry label='Data Entry Time'
format=datetimel3.
informat=datetimel3.;
attrib center label='Site Code'
length=$2.
format=$cntfmt.;
attrib scrnno label='Screening Number'
length=$3.
informat=$3.;
attrib scrmno label='Screening Number'
length=$3.
informat=$3.;
retain numeric
char - -
run;
```

Attrib statements were used in the DATA step to associate each variable with its name, label, length, informat, and format. At the end, a retain
statement was used to set variables default values in an empty data set, missing (.) for numeric variables and blank ('') for character variables.

**Setting Up Data Entry Screens**

After creating data sets, you can use SAS/FSP PROC FSEDIT with a specified screen catalog. This gives you maximum control over data entry screens (Figure 4).

```sas
proc fsedit label data=ecg.ecg2
  screen=ecg.ecg2;
run;
```

**Using SCL to Enhance the Data Entry Capability**

By selecting option 3, edit program statements and compile under FSEDIT MENU (Figure 5), you can program in SCL. SCL can give you more control over data entry, such as automatically fill and validate data values. Please note, Arrays are defined before init section in SCL. Here is a sample program.

```sas
array sc{7} $ sc2 sc3 sc4 sc5 sc6 sc7 sc8;
array sn{7} $ sn2 sn3 sn4 sn5 sn6 sn7 sn8;
array ventcon{6} $ ventcon1 ventcon2 ventcon3 ventcon4 ventcon5 ventcon6;
array vcd{6} $ vcd1 vcd2 vcd3 vcd4 vcd5 vcd6;
```

**Figure 4. Customizing data entry screen**

You can use MOD command under the FSEDIT screen and select option 2, screen modification and field identification under FSEDIT MENU (Figure 5). Temporary variables can be created as read only variables and retained in different screens to facilitate data entry. After designing data screens, you can add some simple data checking capability to the screen by selecting option 4, assign special attributes to fields under FSEDIT MENU.

**Figure 5. FSEDIT menu**
if vcd(i)='l-1' then do;
do j=1 to 3;
st(j,i)='NC';
t(j,i)='NC';
end;
if qsl(i)='2-3' | qsl(i)='2-7' | qs3(i)='3-6' | qs3(i)='2-8'
then do;
qsl(i)='NC';
qs2(i)='NC';
qs3(i)='NC';
end;
......
return;

term:
lastmod=datetime();
return;

CREATING MULTI-LAYER MENU

By using SAS/AF PROC BUILD, you can create multiple layers of menu screens. For example, you can create a program, coordin.program, that can be called by password.program through display() routine. To do that you can type at the command line of BUILD: DIRECTORY ECG.MAIN (E) screen (Figure 1):

edit coordin.program

Then you can edit the screen and set attributes of fields so that the screen looks like Figure 6. The SCL program under BUILD: SOURCE COORDIN.PROGRAM (E) is presented below. A control statement in init section specifies that the main section of SCL will be executed when a ENTER key is pressed. Function curword() returns the character value where the cursor is located and ENTER key is pressed.

init:
    control enter;
a='A';
......
d='D';
return;

main:
    curloc=curword();
    if curloc='A' or modified(a)
    then call display('parta.program');
......
else
    if curloc='D' or modified(d)
    then call display('partd.menu');
......
return;

TERM:
return;

CREATING MULTI-LAYER MENU

By using SAS/AF PROC BUILD, you can create multiple layers of menu screens. For example, you can create a program, coordin.program, that can be called by password.program through display() routine. To do that you can type at the command line of BUILD: DIRECTORY ECG.MAIN (E) screen (Figure 1):

edit coordin.program

Then you can edit the screen and set attributes of fields so that the screen looks like Figure 6. The SCL program under BUILD: SOURCE COORDIN.PROGRAM (E) is presented below. A control statement in init section specifies that the main section of SCL will be executed when a ENTER key is pressed. Function curword() returns the character value where the cursor is located and ENTER key is pressed.

init:
    control enter;
a='A';
......
d='D';
return;

main:
    curloc=curword();
    if curloc='A' or modified(a)
    then call display('parta.program');
......
else
    if curloc='D' or modified(d)
    then call display('partd.menu');
......
return;

TERM:
return;

GENERATING REPORTS FOR TRACKING

Report generating feature can be built on multi-layer menu screens. Under one of the program menu, you can write SCL programs to print reports. One of the reports in the system is to print a list of patients whose ECGs were analyzed within a time period. The SCL codes are the following:

init:
    return;
main:
    /******* Print out for all ECGs with data entry time BETWEEN user input time T1 and T2 ***********/
    IF T2 NE . AND T1 NE . THEN DO;
    submit immediate;
    dm 'output;clear' output;
    data one;
    set ecg.ecg1;
    format datel date7.;
    if coderil ne ' ' and datel ne .;
    IF datel >= "&T1"d and datel <= "&T2"d;
    if newqmi='Y' or newqitm='Y';
    run;
    data two;
    set ecg.ecg2;
    format datel date7.;
    if coderil ne ' ' and datel ne .;
    IF "&T1"d <= datel <= "&T2"d;
    if newqmi='Y' or newqitm='Y';
run;
data three;
merge one (in=i) two (in=j);
by sty_no;
if i or j;
run;
proc print data=three label;
var sty_no date1 coder1 qm1 qtm1
date2 coder2 qm2 qtm2;
title1 "List of patients with Q wave MI or Q wave items";
title2 "All Q waves coded between $T1 and $T2";
run;
dm "output;zoom on;top;" output;
endsubmit;
return;

term:
return;

Sometimes, users want to print data in form of the data entry screen. To print the data by observation or patient and with the same format as screen, you can use PROC FSEDIT with PRINTALL option. The SAS codes are the following:

libname library '../ecg/basessd';
libname ecg '../ecg/basessd';
options ls=80;
proc fsedit label data=ecg.ecgl
    screen=ecg.ecgl printall;

RUNNING THE SYSTEM

You can start the system by running autoexec.sas. The file contains a DM statement which issues the AF command to execute the Automated ECG Coding System. The DM statement tells SAS to start AF from password.program under ECG.MAIN catalog. The AF at the end is to specify the active window is AF window.

libname library '../ecg/basessd';
libname ecg '../ecg/basessd';
dm 'af catalog=ecg.main.password.program' af;

Users can initiate the system by typing SAS at the DOS prompt under directory \ecg where the autoexec.sas is stored.

LIMITATIONS

There are several limitations with this system. As the number of data entry screens, SCL codes, and observations in data sets increase, you need more memory in your PC. A minimum of 512KByte base memory and 2MByte expanded memory is required.

The more the better. Sometimes I have to increase the base memory up to 600KByte and expanded memory to 4MByte. This means you have to compromise PC configuration. Because of SAS version 6.04 SCL codes are limited compare to Mainframe SAS 6.07, UNIX SAS 6.09 and Windows SAS 6.08, some of the SCL codes showed here could have been written more efficient if a newer version of SCL were used.

CONCLUSION

SAS/AF and SAS/FSP are powerful tools to build a user-friendly data entry application. The automated ECG coding system was built in early 1993. As of July 1994 the system has recorded 15,600 ECGs. The system has proven reliable and functional.

References

SAS, SAS/AF, and SAS/FSP are registered trademarks of the SAS Institute, Inc., Cary, N.C.

Acknowledgements

Long Ngo, Ischemia Research and Education Foundation (IREF), provided help to develop the system. Alexander Gager, IREF, maintains the system. Other members of IREF provided suggestions to improve the system. Steve Wilson, Kaiser Permenante Division of Research, encouraged me to present this paper.

Author's Address

Elizabeth Li
Kaiser Permanente, Division of Research
3505 Broadway
Oakland, California  94611
(510) 450-2167

WUSS 1994