ABSTRACT
New releases of SAS software provide greater flexibility for the application developer. No one can argue the merits of moving to Screen Control Language (SCL) from the triple pound macros found in Release 5.18. This time, the transition from program entries to FRAME entries requires modification to both user screens and SCL source code.

This paper is intended for the developer supporting applications in Release 6.06 who wish to take advantage of what Release 6.08 and later have to offer. This paper focuses on moving a MENU entry to FRAME icons, changing a BLOCK program entry to push buttons, and moving from drill-down EXTENDED tables to multiple extended tables on one screen.

INTRODUCTION
After converting a few applications to FRAME entries from PROGRAM entries, it was apparent that some things were very easy to convert and others were very difficult. I say "convert" because there is not PROC PGTOFR like there was a PROC V5TOV6. Some menus and program entries required a complete redesign and others hardly anything. The two main things covered in this paper are moving a menu from a menu entry to a frame menu and a program entry that had 1 extended table to a frame entry that can have multiple extended tables on one frame or window. Menus were fairly straightforward. Extended tables require more effort and more code. You may wonder why would any one want to do this in the first place? As more and more products and applications are going GUI (graphical user interface) oriented, we want SAS applications to have that "windows" look and feel. Often it is said "if it isn't broke don't fix it", but sometimes that can be counter-productive and our users think that because we are using SAS we can't have that GUI interface.

MENU BUILDING
Menu entries tend to be used in character terminals, and are really in need of update considering the GUI interfaces available. Consider the following main menu:

This menu is associated with attributes. The attributes window points to three other program entries:

The MENU entry is easy to 'program' but is limited. For example, there is no way to change the display to the user by program. On the other hand, if the application is to be implemented on a wide range of character terminals, it is one of the most stable entry to use.

PROGRAM / BLOCK ENTRIES
Program Entries can be used to replace menu entries to enhance the visual effect to the end user. In release 6.06 an SCL routine called BLOCK provided a way of displaying a selection of icons. A simple application that replaces the menu entry above now looks like:
This main menu built in a program entry requires associated SCL. Even with only three choices available, the SCL is quite cumbersome. The BLOCK function, for example, requires that choice 11 parameters be passed. Some of these are nulls. The ICON displayed is programmed by number, which is referenced in Technical report P-216(1).

```
INIT: RETURN;
MAIN: DO WHILE (CHOICE NE 0):
  IF (.STATUS_ = 'E' OR .STATUS_ = 'C') THEN GOTO TERM;
  CHOICE = BLOCK ('BORDER HEADER',
      'MAINMENU TITLE',16,
      'CHOICE 1',
      'CHOICE 2',
      'EXIT,
      ' " " " " " " " " " " " " " 
      ,231,111);
SELECT (CHOICE);
  WHEN( 1 ) CALL DISPLAY('CHOICE1.PROGRAM');
  WHEN( 2 ) CALL DISPLAY('CHOICE2.PROGRAM');
  WHEN( 3 ) CALL DISPLAY('EXIT,PROGRAM');
  OTHERWISE;
  END;
END;
CALL ENDBLOCK();
RETURN;
TERM: RETURN;
```

The SCL is also more maintainable. Each screen object has its own SCL section.

```
CHOICE1: CALL DISPLAY('CHOICE1.PROGRAM'); RETURN;
CHOICE2: CALL DISPLAY('CHOICE2.PROGRAM'); RETURN;
EXIT:   CALL DISPLAY('EXIT,PROGRAM'); RETURN;
```

**EXTENDED TABLES**

We have seen the simplicity of using FRAMES for building menus. Extended Tables are one of the more difficult SAS / AF applications to build. FRAMES however, provides a great deal of flexibility beyond extended tables in program entries.

Review of extended tables in program entries:

```
CREATE PROGRAM ENTRY OF EXTENDED TABLE
```

**FRAME ENTRIES**

FRAME entries provide yet another enhancement to the main menu by allowing the application builder a wider range of 'Objects' displayed. In the above menu application, the BLOCK SCL that generates each all icons is replaced by the creation of objects directly on the frame with the ACTIONS...MAKE...ICON pull down menu. Each Icon is placed in a 'REGION' (with a different color) and a GRAPHICS object is used for the title. The net result is a considerable enhancement to the program entry:
The SCL for this program entry is

```
INIT:
  DD=OPEN('SUGI.CITIES');
  CALL SET(DD);
  CALL SETROW(10);
  RETURN;

MAIN:
  RETURN;

TERM:
  DD=CLOSE(DD);
  RETURN;

PUTROW:
  RETURN;

GETROW:
  IF FETCHOBS(DD_CURROW) < 0 THEN CALL ENDTABLE();
  RETURN;
```

A major limitation of extended tables in program entries is that only one extended table can appear on a screen.

**Extended tables in FRAME entries**

We will start with one extended table on the frame and then add another extended table which is linked to the first extended table. You create extended tables by starting the Build procedure:

```
PROC BUILD c=SUGI.SUGI.EXT.FRAME;
```

The popup menu ACTIONS .. MAKE .. EXTENDED TABLE will display an outline of the extended table. You need to place the extended table outline on the frame. Once placed on the frame, the attributes window for the extended table will appear. You have several options to name the extended table (default=OBJxx) and corresponding GETROW and PUTROW section.

For this example, the extended table is set as follows:

Name:    EXT1
GETROW:  GET1
PUTROW:  PUT1
DYNAMIC TABLE ON
Horizontal SCROLL OFF
Vertical SCROLL ON
SELECTION LIST ON
ROW SPACING 0

You press OK when done. This will return you to the frame with the outline of the extended table:

You then need to make a text entry field using the ACTIONS .. MAKE .. TEXT entry and place it in the extended table:
The attributes for this text entry field are:

You press OK to display the extended table with its text entry field inside. Use the LOCALS..EDIT SCL SOURCE popup menu to enter the SCL program that will run the extended table. In this case, a very simple data set is accessed with an OPEN call and the GET1 section populates the extended table:

INIT:
DD1=OPEN(,SPUG.CITIES·,T);
CALL SET(DD1);
RETURN;

MAIN:
RETURN;

TERM:
DD1=CLOSE(DD1);
RETURN;

GET1:
IF FETCHOBS(DD1~CURROW..J<0 THEN DO;
  CALL NOTIFY('EXT1','_ENDTABLE_');
END;
RETURN;

PUT1:
RETURN;

You compile the program using the LOCALS..COMPILE, and it will be ready to be used. The final display, with the data looks like:

A number of enhancement can be done to this table. For example, the spacing between lines can be reduced by sizing the text container outline, color can be changed on text and more.

TWO EXTENDED TABLES

The following example shows how two data sets, related by the variable CITY can be displayed. The first extended table has attributes:

Name: EXT1
PUTROW section: PUT1
GETROW section: GET1
Variables: CITY

The second extended table has the following attributes:

Name: EXT2
PUTROW section: PUT2
GETROW section: GET2
Variables: NAME COMPANY

The display is:

The SCL behind this application is:

INIT:
DD1=OPEN('SPUG.CITIES','T');
DD2=OPEN('SPUG.NAMES','1');
CALL SET(DD1);
CALL SET(DD2);
RETURN;

MAIN:
RETURN;

TERM:
DD1=CLOSE(DD1);
DD2=CLOSE(DD2);
RETURN;

GET1:
IF FETCHOBS(DD1,CURROW..J<0 THEN DO;
  CALL NOTIFY('EXT1','_ENDTABLE_');
END;
RETURN;

PUT1:
RC=WHERE(DD2,'UNDO');
RC=FETCHOBS(DD1,CURROW..J;
SELNAME=CITY;PUT SELNAME=;
IF SELNAME NE " THEN DO;
  PUT SELNAME=";
  W="CITY="" ! SELNAME !! "";
  RC=WHERE(DD2,W); PUT RC=;
  CALL NOTIFY('EXT2','_UPDATE_');
END;
RETURN;

*-------------------------------------------------------------------;

GET2:
  LINK SELECT;
  RC1=FETCHOBS(DD2,CURROW->);
  IF RC1 = -1 THEN DO;
    CALL NOTIFY('EXT2','_ENDTABLE_');
  END;
RETURN;

PUT2:
  LINK SELECT;
  RC=FETCHOBS(DD2,CURROW->);
RETURN;

SELECT:
RETURN;

The first extended table behaves as a selection list. When a particular row (CITY) is selected, a WHERE clause is applied to the second extended table (EXT2). For example, selecting 'Calgary' results in different names displayed on the second extended table:

Recommendation - take the plunge and upgrade your applications to GUI. Remember the user comes first and they will expect to have that "windows" look and feel. No, there is no PROC to do the work, you may end up redesigning or re-engineering your processes but it will be worth it in the end.

REFERENCES

2. SAS/AF Software: FRAME Entry Usage and Reference, Version 6, First Edition
3. SUGI 19 Proceedings: Getting Intimate with Extended Tables

SAS, SAS/AF are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. © indicates USA registration.

FOR MORE INFORMATION

Loretta Golby
ISM Alberta,
Edmonton, Alberta
(403) 945-4722

Serge Dupuis
BKD Software Consultants
Edmonton, Alberta
(403) 466-4882

CONCLUSION

SAS is keeping up with the times and we need to keep our applications up to date. The further behind that any application becomes the harder it is to use, maintain and upgrade to new releases of the software. FRAMES provides for more flexibility but more training or experience with SCL is required.

FRAME menus are just one way to enhance your applications from traditional text based menu screens to GUI menus. FRAME menus are easier to maintain but not extended tables.