Paper 239

The Use of DATA FORMS, TABLES and other new OBJECTS in SAS/AF®

Bernd E. Imken, Patented Medicine Prices Review Board, Government of Canada, Ottawa, Canada

Abstract:

This paper was originally presented to the Ottawa SAS Users Group. When the audience was polled as to the methods of data access they were using, it was discovered that only a small percentage used SAS/AF. The Patented Medicine Prices Review Board had been using AF including Data Forms and Data Tables successfully for some time. The purpose of this paper is to inform you as to how you too could migrate to this newer technology. Other objects which are also discussed, because of their usefulness, include - the TOOLBOX object, the TAB LAYOUT object, and the PROCESS FLOW DIAGRAM object. One additional topic also referred to is the ability of sending system eMails from a data step.

Introduction:

The Patented Medicine Prices Review Board regulates the maximum prices charged on patented medicines sold in Canada. Currently patentees sell nearly $4 Billion each year. A more complete description of the PMPRB and it's mission may be seen in the Sugi 24 paper - *"Improving SAS Data Access - An Evolutionary Experience".*

The PMPRB data base contains in excess of 8 million SAS observations. All users navigate through the system using screens developed completely as SAS/AF Frames. When you click on the SAS95/PMPRB icon in the Windows environment you are transported directed to the application. This was accomplished using the following command within the CONFIG.SAS

-initcmd "AF c=PMPRB.PROD.OPENING.FRAME"

When this command is processed by the SAS configuration, SAS proceeds directly to the first frame without giving you access to the Program Manager or Log. Such an approach is excellent when you have no or little SAS programming knowledge. Figure 1 displays a frame filled with a TAB OBJECT. By selecting the corresponding TAB, you can either access the PMPRB data base or other data bases on Tab2.

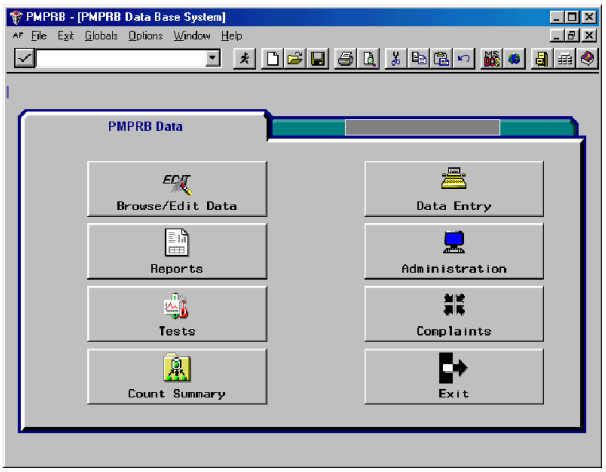

Figure 1 Main Menu

If you click on the first IMAGE ICON, called Browse/Edit Data, you are immediately passed to the screen below.

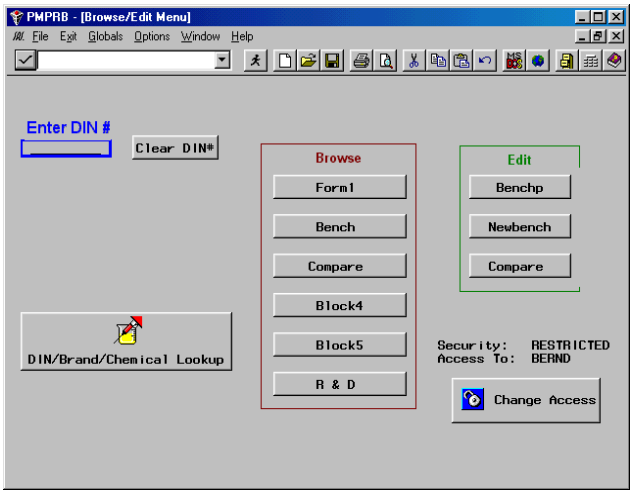

Figure 2 Browse/Edit Menu

This screen contains further buttons which you select to go to any part of the data base.

When you click the Bench button you go to the screen which is shown below. This screen is made up of a multiple page DATA FORM on the first TAB and additional linked DATA TABLES on the next TABS, as is illustrated in Figure 4.
Most screens use customized PMENUs, TOOLBAR objects and other AF objects including GRAPHIC TEXT,

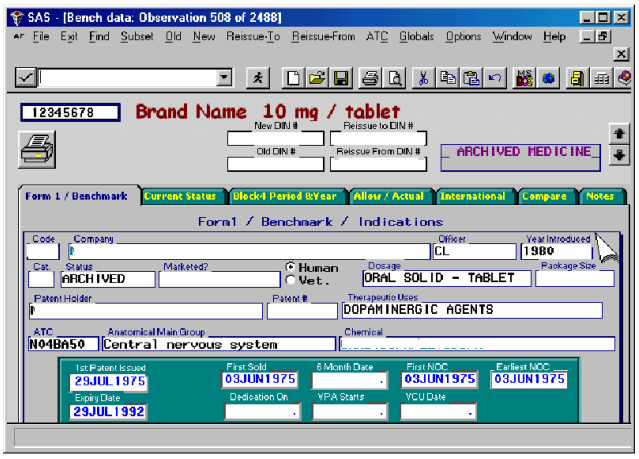

Figure 3 Drug Product Benchmark Data

and TEXT INPUT objects as is shown in Figure 4 on the next page.

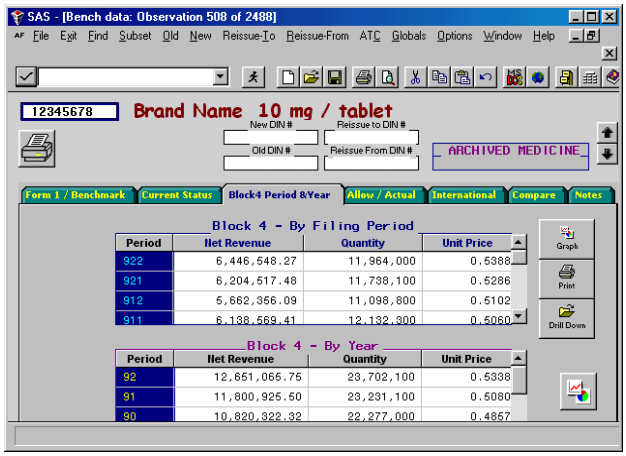

Figure 4 Sales by Period & Year

End users at the PMPRB have found this system to be easy to use, even if they have absolutely no knowledge of how to use SAS.

Now Let's Build Our First DATA FORM

You Must first enter SAS/AF by typing:

PROC BUILD C=sasuser.afdemo;run;

Once this is done enter a blank FRAME screen by typing:

EDIT mydemo.frame

In this small simplified example, I will use the small data set containing only a salesman's name and the quantity of sales he achieved.

The "mydemo.frame" opens with a blank screen. Mark an area on the screen in which you want to place an object. When one chooses ACTION and selects FILL, you will be presented with listing of objects which are available from

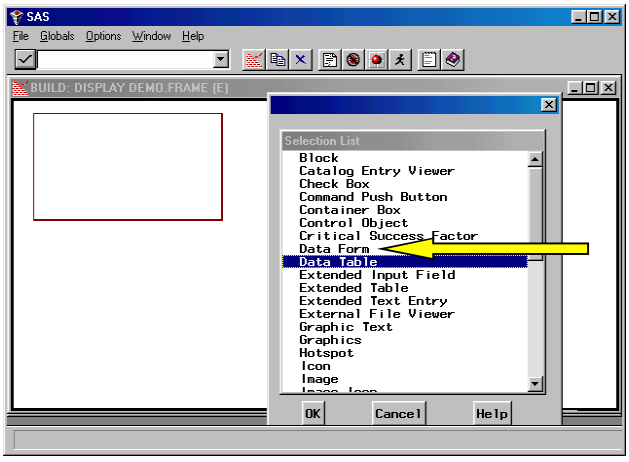

Figure 5 Filling a marked area with a Data Form

SAS/AF. Note, this process to set up the DATA FORM is exactly the same as setting up any other SAS AF object. For those of use unfamiliar with this approach, there are excellent SAS courses which address these concepts. Once the Fill Selection box appears select Data Form.

Once DATA FORM has been clicked the next screen will open up automatically.
The DATA FORM ATTRIBUTES screen(same as DATA TABLE screen) - also called the OBJECT ATTRIBUTES,

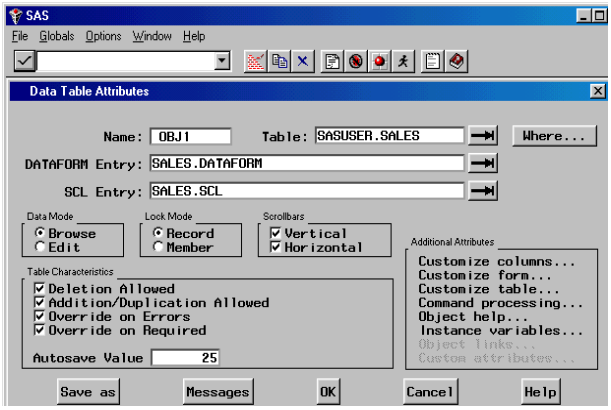

Figure 6 Data Form, Object Attributes Screen

is the place where you specify the data set name which will be viewed. Here, the data set name is SAUSER.SALES, this is recorded in the TABLES box. This is the one mandatory field which must be specified to bring up a DATA FORM. If only this were to be input the default data form would be created by SAS but you would not be able to store modifications to the form to approve usability or appearance. In order to store the screen design there must be an entry into the DATAFORM ENTRY field, here I inserted SALES.DATAFORM. Similarly if processing code is maintained with the screen the entry is specified as an SCL entry.

You can set as many options on the screen at this time as you desire, but you may also exit the screen and go directly to the generated, or default screen by pressing F3. You may return to the OBJECT ATTRIBUTES screen at any time to make further changes by right clicking your mouse on the object and then selecting OBJECT ATTRIBUTES. In figure 7 you will see the default screen designed by SAS.

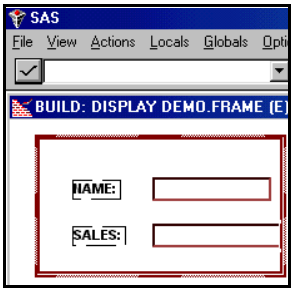

Figure 7 Default Data Form

This screen is now fully useable, allowing you to navigate through the data set. So far the process has been about as easy as PROC FSEDIT. Let's go on.

Now let's try to improve the design of our screen as in figure 8. Let's say that you also want to place on a title, go to the default screen and mark another area on the screen, select the right mouse button and click on

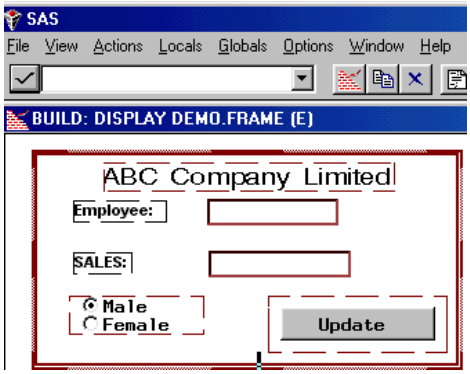

Figure 8 Customizing a Data Form

GRAPHIC TEXT. On the screen provided for graphic text type in “ABC Company Limited”. In a similar fashion you may insert radio buttons, image icons, or any other AF Object. At this stage the process is really no harder than designing an FSEDIT screen - but you have much more graphical control. In addition, a radio box with two buttons one for “Male” and one for “Female” would correspond with a data set also having a variable with the same values. Features like this were never available in FSEDIT.

Note: if an object had been developed after the initial development of the default screen, then it would have been necessary to link the data set variable with the screen variable. To do this first right click on the object, the following screen will be displayed.
Instead of clicking on OBJECT ATTRIBUTES, this time select FORM ->

This will display another menu with DISPLAY COLUMNS at the top. When this is selected a list of all the variables in the data set will be displayed. Select the variable you wish to link, in this case NAME. Drag it to the DATA FORM screen and drop it on the newly created radio button . The screen will display that there is a linkage at the bottom message line.

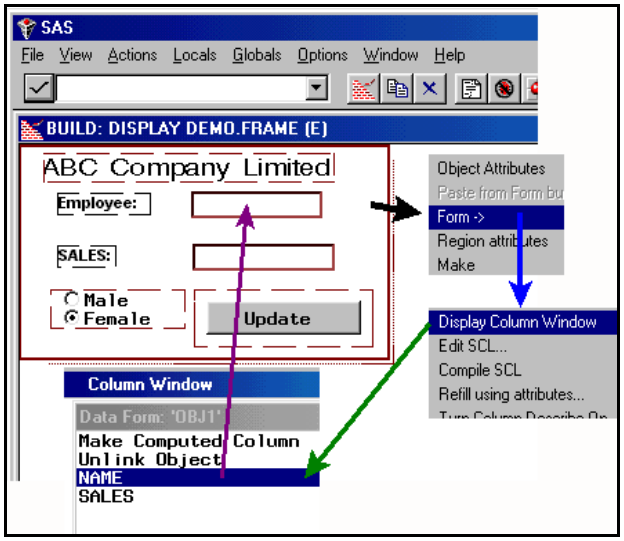

Figure 9 Linking a SAS variable to a screen object

Now let’s modify the screen even further. First let’s go back to the OBJECT ATTRIBUTES screen. Remember this is always done by clicking the right mouse button while positioned over the object.

Let’s look a bit closer at the OBJECT ATTRIBUTES screen. This is displayed at the top of the next column.

If you select the customize columns attribute on the right hand side of the screen another window is displayed. This screen allows you to specify which data set variables - also called columns, you want displayed on your screen. Notice at the bottom of the screen you could also add screen variables which do not become part of the data set, these are similar to computed variables in the FSEDIT world.

Next you could select FONTS from the left column and the

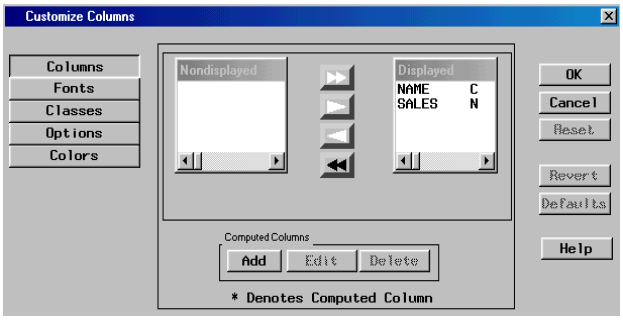

Figure 10 Modifying Fonts, Displayed Variables, Min., Max etc.

follow screen would be displayed allowing you to change font sized, types and colors. You could also have selected OPTIONS and the following screen would be displayed. This screen allows you to select MIN MAX, etc. just like you could in FSEDIT.

The OBJECT ATTRIBUTES screen also allows you to customize your FORM. Select the Customize Form option. SAS will now display a screen allowing you to select background color etc. You may also wish to see your labels as variables name instead, or pair labels and boxes for easier control. Specifying a KEY Column also allows you to link the data form you are creating with another data form or data table.

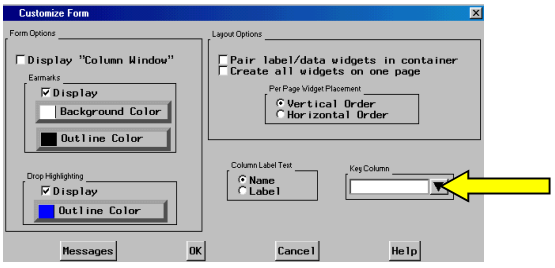

Figure 11 Specifying a Keyed Variable

Entering SCL associated with this data form is also very easy but THERE IS ONE IMPORTANT THING TO REMEMBER. The DATAFORM has it’s own SCL as was specified in the OBJECT ATTRIBUTES.

To access the DATA FORM SCL right click on the object again and select “FORM ->” When the menu is displayed select EDIT SCL. It is very important to realize that this is not the same as selection EDIT SCL from LOCALS at the

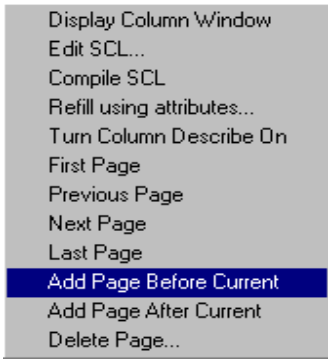

Figure 12 Edit SCL/ Modify Screen

top of the screen. The SCL at the top of the screen applies to the entire frame and does not communicate directly with the SCL from the DATAFORM. Once your DATAFORM SCL has been entered you may press the compile button (compile does not happen automatically as it does using FSEDIT)

Once you are ready back out of the attribute windows by pressing the F3 key and return to the new screen which you have developed. Like in any FRAME object which you develop, all the objects can be RIGHT Clicked or dragged, or resized.

When you are done select TESTAF from the LOCALS menu item or submit the following command to the command line:

AF c=sasuser.afdemo.mydemo.frame

Your newly designed screen is now usable.

Figure 13 shows the first page of a data form, the second page is shown on the right.

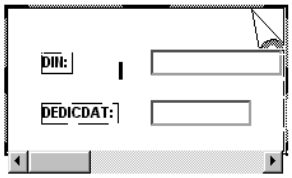

Figure 13 Screen #1

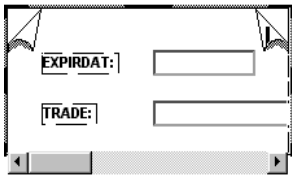

Figure 14 Screen #2

Clicking the page curl at the top right or left allows the user to navigate between pages.

Hint: If you are dealing with a large DATA FORM which will span many pages you might find the following useful. Instead of dealing with moving variable and labels from page to page in a form, I have found it personally more useful to first design a screen with only a limited number of variables (1 or 2). Then expand the size of your data set by populating it with new variables. Then bring up the variable selection window to allow you to drop and drag the new variables to their desired location.

Hint: If you add a push button, icon, or image icon, it must be added to the screen as a computed item with the same object name as a SCL Section.

So far the it can be seen that although it is different than using FSEDIT, it is really not much harder. The same thing can be said for creating a DATA TABLE. Once the area is marked on your frame screen, select DATA TABLE from the FILL options.

You will be moved automatically to the DATA TABLE ATTRIBUTES screen which looks the same as the OBJECT ATTRIBUTES screen which you had just been using to develop a DATA FORM.

The only difference with a DATA TABLE is that you can also click on customize columns on the OBJECT ATTRIBUTE screen to control whether you want to see row and column labels, grids etc.

If you want to enter SCL for your table click TABLE-> rather than FORM-> as you did before.

Now a More Complicated Example:

So far in the development, the process has been as easy as doing an FSEDIT screen. With DATA FORMS and DATA TABLES, however, you can do so much more. One good example is when you would like to display a MASTER data form and its associated TRANSACTIONS on a separate data table on the same screen.

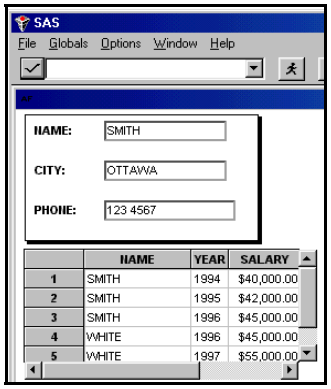

Figure 15

In Figure 15, I have created a Data Form and a Data Table on the same FRAME.

The master information is on the top - the data you want to associate is on the bottom of the screen. In the bottom data table, however, you only want to see those transactions which match with 'SMITH" above.

In order that this can be done you must write SCL for the frame to identify the data form and the linked data table. you also write SCL code for each of the Data Form and the Data Table.

The SCL code for the entire FRAME is displayed below:

```
length w $20;
INIT:  control always;
        call send( _frame_, '_GET_WIDGET_', 'EMP', dfid );
        call send( _frame_, '_GET_WIDGET_', 'SALARY', tabid);
          call send( dfid, '_SET_EVENT_HANDLER_', dfid,
                'DF_ROW_CHANGED', '_SET_KEY_', tabid);   RETURN;

MAIN: control always allcmds;
        w = word(1,'u');
        if w ='BACKWARD' then do;
            call send( dfid, '_VSCROLL_', 'row', -1 ); end;
        if w = 'FORWARD' then do;
            call send( dfid, '_VSCROLL_', 'row', 1 ); end;        RETURN;

TERM:       RETURN;
```

The SCL tells us that the data set for the MASTER is called EMP. The INIT section identifies this to the frame as DFID. Similarly the data set for the TRANSACTION is called SALARY as is identified as TABID.

The other line of code in the FRAME tells the frame that when a row is changed in the MASTER, change to the linked or KEYED rows in the TRANSACTION file.

Below you will see the SCL code required for the DATA FORM.

```
DFINIT:  CONTROL ALWAYS;
            vals = makelist();       RETURN;

INIT:  CONTROL ALWAYS;
             rc=clearlist( vals );
             rc=insertc( vals, NAME, -1, 'NAME' );
             call send( _viewer_, '_SEND_EVENT_',
             'DF_ROW_CHANGED',  rc, 'NAME', 'EQ', 'SCROLL', vals);
RETURN;

DFTERM:
          rc = dellist( vals );          RETURN;
```

The call send tells the data table (_viewer_) that when the variable NAME changes on the DATA FORM the data table should call up transactions equal to the new NAME being displayed. Notice that the NAME variable must be the KEY column on both the data form and the data table. Also NAME should properly be indexed in both files for performance reasons.

Now let's look at the code for the Data Table:

```
DFINIT:
          call send ( _viewer_ ,  '_DISPLAY_COLUMN_LABEL_', '_ALL_' );
         RETURN;
```

The only purpose for this code is to tell the DATA TABLE to display the column labels.

The MASTER / TRANSACTION frame should now be ready (after all three compiles) for testing. The result is show in figure 17.

Additional code could be added to the FRAME to permit multiple label lines to be shown. An example of the same is shown at the below.

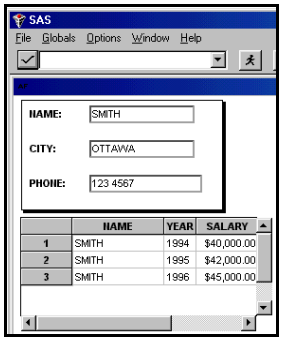

Figure 16 Data Form linked to Data Table

```
odelid=getnitemn(tabid,'DATAID');

  call send(modelid,'_set_instance_method_','_get_column_info_',
          'dev.newbench.splitlbl.scl','getcoli');
  call send(modelid,'_set_instance_method_','_get_column_dim_info_',
          'dev.newbench.splitlbl.scl','getcdimi');
  call send(tabid,'_set_dataset_',' ');
  call send(tabid,'_set_dataset_','pmprb.benscr2','browse');
  call send(tabid,'_display_column_label_','_all_');
```

This code requires the mothod to be stored as SCL code as shown below:

```
getcoli:
  method rcdvecid 8;
    call super(_self_,'_get_column_info_',rcdvecid);
    call send(rcdvecid,'_set_wrapping_','y','*');
    call send(rcdvecid,'_set_vjust_','bottom');
endmethod;

getcdimi:                                                    (Continued..)
  method coladdr elements subdim height 8 units $ 5 grouplst 8 eod $1;
  call super(_self_,'_get_column_dim_info_',
              coladdr,elements,subdim,height,units,grouplst,eod );
    if (listlen(coladdr)=0) then
    do;
      units='fit';
    end;
endmethod;
```

One can see that the level of SAS SCL coding rapidly becomes more complex. Generally once you are this level and you really want to get additional power out of DATA FORMS and TABLES, experience SCL programmers with a solid background in SCL list , object, classes and methods is required.

Now let's look at other Useful Tools

First let's look at the TAB LAYOUT OBJECT. This one is easy to use and develop. Most object which can be placed on a frame can also be placed on a TAB object. The beauty of this is that it permits access to many things on the TAB FRAME without appearing cluttered.

Earlier, you saw that even DATA FORMs and DATA TABLES can be placed on a TAB OBJECT. To set up a TAB, simply mark the area for the TAB and FILL with the

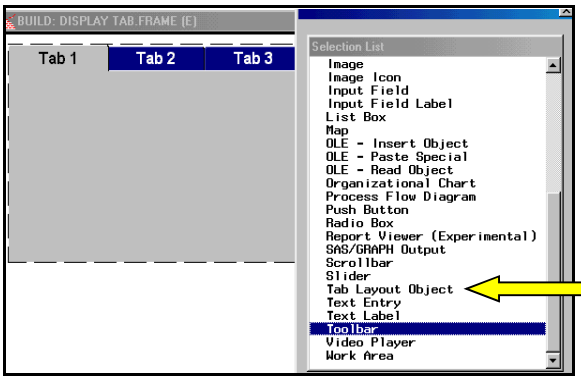

Figure 17

TAB LAYOUT OBJECT. Click next on OBJECT ATTRIBUTES to change the number of folders etc.

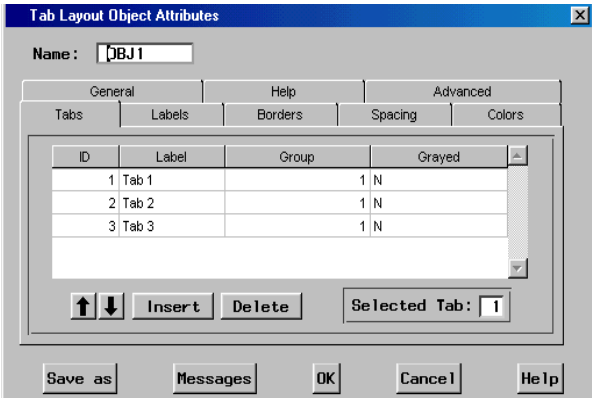

Figure 18

The rest of the process is simple. Simply refer to the OBJECT name you created on the TAB folder just as though it existed directly on the FRAME itself.

Another useful OBJECT in AF is the Process Flow Diagram - it's great for doing system flowcharts etc. Figure 20 is an example.

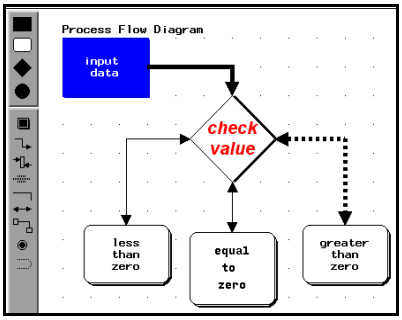

Figure 19 Process Flow Diagram

Here the developer or user only needs to drag and drop from the bar on the left. Object created can be resized etc. text can be added. Arrows can be added. Arrows remain linked even if an object is later moved. It's a great little tool with a few good applications.

Another little used OBJECT is the TOOLBAR object. It is used extensively at the PMPRB rather than using individual icons. An example of it's setup is shown below, followed by an example of a toolbar set up with the alphabetic to aid in searches.

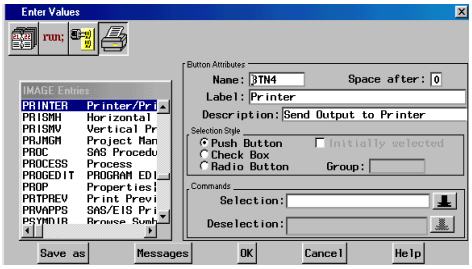

Figure 20 Enter Data to Toolbar Object

Providing tools such as these has gained widespread acceptance at the PMPRB and are used heavily.
One last tool which the PMPRB uses is the ability of SAS to communicate through eMail. The PMPRB uses MAPI and finds SAS communicates well with it. Messages can be sent by clicking SEND on the FILE menu. In this case an template is provided for the message as below.

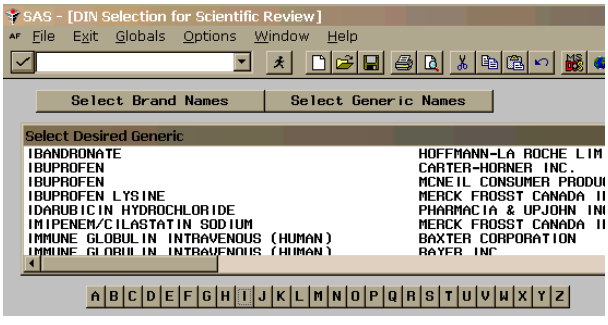

Figure 21 Alpha Toolbar to assist drug selection

In order to use this facility the following must be added to the CONFIG.SAS

-EMAILSYS MAPI
-EMAILDLG SAS
-EMAILID "Novell Default Settings"

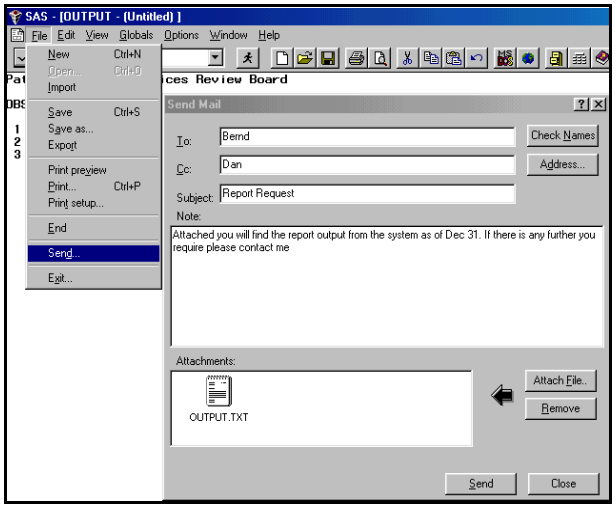

Figure 22 Using SEND with SAS

eMails directly within a Data Step (see Page 30, Microsoft Windows Environment: Changes and Enhancements to the SAS System Release 6.10).

Conclusion:

The effort involved in setting up data forms and tables the first time is considerable ,but the process become much easier after some experience. The result for the end users is well worth it.

References:

The following technical notes are extremely useful.

1. ***An Introductory Overview of the Data Form and Data Table Objects in SAS/AF Frame Entries***
   - Ann Rockett, SAS Institute

2. ***Introducing a GUI Approach to Data Entry Using Data Table and Data Form Objects in SAS/AF Frame Software***
   - Ann Rockett and Lynn Leone, SAS Institute

3. ***Data Table and Data Form Enhancements in Release 6.12***
   - Scott Wilkins, SAS Institute

The author may be contacted at:

Bernd E. Imken
Chief, Information Systems Division
Patented Medicine Prices Review Board
333 Laurier Avenue West
Ottawa, Ontario K1P 1C1
Phone 613 952-3312 Fax 613 952-7626
E-mail bimken@pmprb-cepmb.gc.ca