Graphical User Interface to Report Generation in a Pharmaceutical Environment
(Using SAS/AF® Software FRAME Technology to Keep Your Head Above Water)

Kenneth Salatka, PARKE-DAVIS Pharmaceutical Research, Division of Warner-Lambert Company
Nancy Brucken, PARKE-DAVIS Pharmaceutical Research, Division of Warner-Lambert Company

ABSTRACT
This paper describes the user interface developed at PARKE-DAVIS Pharmaceutical Research. This interface is used as part of a CANDA (Computer Assisted New Drug Approval). The system was developed with SAS/AF® FRAME software. Two important aspects of any New Drug Approval are listing tables that contain all patient information and summary tables that describe the safety and efficacy of the drug. We have developed a reporting system that allows any user to quickly produce and review a production quality report (listing or summary table). This is an object oriented "point and click" system that will execute SAS Base software to produce the desired output.

INTRODUCTION
There are two main components to the reporting system developed at PARKE-DAVIS, the Graphical User Interface (GUI) and the production report generator. The report generator is the more complex component of the system and, therefore, was more difficult to program. The report generator was designed to be very flexible, which made the user interface easier to develop.

One of our drug development programs has 19 different studies, with each study requiring 265 different production reports. A total of 5035 different production reports will be produced for that drug development program. A well organized system for producing these reports is essential. A Graphical User Interface is a tremendous advantage for people who review the clinical trial data. It allows them to run the tables and look at the output on their computer without having to print it. Furthermore, reviewers can use the GUI to select different subsets of data for analysis: a task which was previously difficult to accomplish.

PRODUCTION REPORT GENERATOR
Production report generation is accomplished using SAS® Base software. These reports are placed directly into the research report for the Food and Drug Administration (FDA) without any modification. The reports must be well organized, have a neat appearance and most important, be absolutely correct. The SAS® system does an excellent job of doing all this. It must be stressed that report generation is the most important part of the system. The report generation software was deliberately programmed to interface nicely with a GUI. Good parameter definition allows easy communication between the GUI and the report generation software.

Most summary table programs need to produce reports based on patient type, visit type and patient qualification. All summary table programs must also produce reports by center (place of treatment) and some need to produce reports by diagnosis. Therefore, there are two basic activities that can be made to work with all tables:

1) subset data for patient type, visit type and patient qualification.

2) use a general macro program to organize the output to print each report by center or diagnosis when requested.

A set of routines has been developed to make programming summary tables quick and consistent. There are three basic routines:

1) A string parsing routine that sets global macro variables for use later in the program.

2) A subsetting routine that produces a data set from the original without altering the original data set.

3) A routine that builds SAS® code into macro variables, which are used to produce tables by center and diagnosis.

STRING PARSING
The first routine, called RESTPAR, will parse a SAS® macro variable to get user input. When a program is executed the possible parameters can be entered as follows:

program_name///CBE,TOC,DIAG,CEN,QCB,ASSOC
Where CBE = Patient type, TOC=Visit type, DIAG=by diagnosis, CEN=by center, QCB=patient qualification and ASSOC=associated. Not all of these parameters are necessarily entered at the same time. This list simply shows possible combinations. The three slashes after the program name will cause the user input data to be placed into a macro variable called JSMREST. The RESTPAR macro will parse JSMREST and then set six global macro variables: 

- **pttyp** - patient type,
- **vst** - visit type,
- **bydiag** - by diagnosis,
- **bycent** - by center,
- **qutyp** - patient qualification,
- **aetype** - adverse event type.

These variables are available for use by any routine after RESTPAR is called. The valid values for the variables are:

### **pttyp**
- CBE (Clinically & Bacteriologically Evaluable)
- CLiN (Clinically Evaluable)
- MITT (Modified intent to treat)
- ITT (Intent to treat [Default] [All Patients])

### **vst**
- TOC (Test of Cure)
- LT (Long-term Follow-up)
- blank (Everything [Default])

### **bydiag**
- DIAG (By diagnosis)
- blank (across all diagnosis [Default])

### **bycent**
- CEN (By Center)
- blank (across all centers [Default])

### **aetype**
- ASSOC (Only associated Adverse Events)
- blank (all AE's)

### **qutyp**
- DQCB (Disqualified clinically and bacteriologically)
- DQC (Disqualified clinically)
- QC (Qualified clinically and bacteriologically)
- QC (Qualified clinically)
- blank (all patients)

If RESTPAR finds a string that does not "match its valid list of strings then an error message is printed in the output and the SAS' job is aborts.

### SUBSETTING THE DATA

The second macro routine produces a temporary data set that is a subset of the original data set. This routine is called REPSUB. The **pttyp** and the **vst** variables are used to perform the subsetting. A data set name must be passed to the macro routine. This data set will be used to create a new and smaller data set. When producing listing and summary tables for review, usually only one SAS' data set is analyzed at a time. Occasionally, some merging of other data sets is necessary. Obtaining the initial data set is usually a matter of using a SET statement with some standard subsetting, like getting the desired protocol. This is not a large part of the process but it is important and will only be performed once for each SAS' job. A routine called SETUP gets the initial data set that the program will use. Frequently, several production reports are run at the same time. These reports will almost always require the same initial SAS' data set. Therefore, we only want to create the initial data set once for the entire SAS' job. The subsetting routine will not modify the data set, but create a new one.

If the setup routine was called with CLIN as the desired data set then REPSUB would be called in the same way. For example;

```sas
SETUP(CLIN); /* get initial data set */
RESTPAR; /* get user input parameters */
REPSUB(CLIN); /* subset data based on user input */
```

The original data set called CLIN will not be altered, but a new data set will be created called CLINQ. Another global macro variable called **subs** is created in REPSUB. It is set to blank if no subsetting is performed and set to Q if a subset is created. In our SAS' program the macro variable is always appended to the name of the data set. That way if a subset was created it will be used, otherwise it will use the original data set. Example:

```sas
DATA CLIN1;
SET CLIN&subs;
IF CLASMT=1;
```

The macro variable **subs** can assume two values: blank or Q. If blank, the original data set is used. If **subs** is set to Q, the subsetted data set is used. Thus, the SET statement will have one of two names; CLIN or CLINQ. If a program is executed later in the same job submission and the CLIN data set is used again with no subsetting, SETUP will not have to recreate CLIN.
If a later program requires subsetting, then REPSUB will overwrite CLINQ thereby saving memory space.

**SAS® CODE GENERATOR**

The third macro routine builds pieces of code that are run later in the program (A nice feature of interpretive languages). A routine called BYCENDIA sets a group of global macro variables based on what the user designated for diagnosis and center. These variables contain critical SAS® code to control how the SAS® procedures are executed. Example:

```
PROC SORT DATA=CLIN1;
 &fsortby;
```

&fsortby may contain one of the following SAS® statements depending on the user input:

- `BY PROT RXGRP PTNO;` (1)
- `BY TRIAL PROT RXGRP PTNO;` (2)
- `BY TRIAL DIAG RXGRP PTNO;` (3)
- `BY DIAG RXGRP PTNO;` (4)

If the user wants a summary of the entire study, then the first BY statement would be substituted for &fsortby. If a summary of the entire study by center is desired then, the second BY statement would be used. The third BY statement is for summaries by center and diagnosis, while the fourth BY statement is by diagnosis only.

The BYCENDIA routine is called after RESTPAR has been called. The code in our SAS® program referring to center and diagnosis will have a macro variable to define how to handle center and diagnosis. BYCENDIA has the logic to identify what the user has chosen. This logic will call another macro that simply sets up all the necessary variables to be used in the program.

Another macro program has been developed that will set the SAS® TITLE statements based on user input. This will cause the correct titles to be printed. It uses the four macro variables from the input parsing routine. This routine is called after SETTITLE. The titles are set for visit type, patient evaluability, and diagnosis. Examples of titles are shown here.

These titles describe the visit type for an antiinfective study:
- **Test of Cure Visit**
- **Long Term Follow-Up**

These titles describe the patient type:
- **Clinically Evaluable Patients**
- **All Patients**

These titles describe how the report is organized:
- **Across all Diagnosis**
- **By Diagnosis**

An example of a listing table and summary table is presented here.

![Figure 1 Listing table report.](image1)

![Figure 2 Summary table report.](image2)
SELECTION PROCESS: SAS/AF® Software FRAME Technology
The previous section described the program flow and organization. This section describes the user interface. Making the table selection and choosing patient subsets is very easy for the user. One does not have to remember that CLIN means to retrieve only the clinically evaluable patients, or any of the other acronyms. Using the CUI, everything is presented clearly to the user without confusion. A "point and click" interface was developed to minimize keystrokes.

The user selects a table and the additional parameters. The command line is built for the SAS* job as described in the previous section. The job is then submitted for execution. This approach has worked very well. During development of a particular report the programmer does not necessarily want to use the point and click interface each time the program is to be executed. All the programmer has to do is submit the command line for execution. When the program development is finished, the program can be easily added to the interface as a selection for other users.

The following SAS/AF® FRAME screens show the basic flow. A consistent color scheme was implemented. Red and green can be "clicked". Red always cancels the current activity and takes the system back a screen. Green always makes a selection or initiates a process. Blue is informational and cannot be modified. "Clicking" on blue does nothing.

In figure 3, the user simply "clicks" the mouse cursor on the box indicating the type of report desired. If the Summary Tables box is "clicked", then screen shown in figure 4, will appear. A type of summary table report is chosen from the buttons at the top of the screen. The radio
buttons in the middle allow the user to select patient subsets based on analysis and visit types. If the user "clicks" on the Clinical Outcome, Investigator button, then the screen shown in figure 5 appears, allowing the user to select the actual report. Once the user "clicks" Select Table and Generate Table, the report is produced and displayed to the user. The report can be printed or saved to a file.

Code development to do this is relatively simple. The GUI development package provided by SAS Institute is an excellent tool. Actual screen design and creation for the entire system took about 5 days. The SCL code determines which objects on the screen are activated. Parameters are set up and sent to task that is initiated. The details of this are simple but it would take up a great deal of space to explain them here.

The Screen Control Language manual and the SAS/AF Software FRAME Entry manual explain these concepts clearly. This paper demonstrates the practical application of various types of SAS software.
CONCLUSION
The reporting system developed at PARKE-DAVIS has greatly helped those who have used it. This system will continue to be used and enhanced. GUI interfaces are being increasingly applied in pharmaceutical research. SAS/AF Software is an excellent tool for developing GUI interfaces.

SAS, SAS Base, SAS/AF, SAS/AF Software FRAME Technology are registered trademarks or trademarks of SAS Institute Inc.

Other brand and product names are registered trademarks or trademarks of their respective companies.