The excellent time-handling functions of the SAS base product, coupled with the parameter passing abilities of the MACRO LANGUAGE, enable a superior profit reporting system for transaction-based data in foreign currency trading. The resulting program is significantly more elegant than the PL/I program previously used for this application.

Introduction:

The Foreign Exchange department in banking seeks to earn profits on that portion of the bank's deposited funds which it manages by trading on the ever-varying currency exchange rates existing in the world's money market.

Skilled traders seek to exchange dollars—or promises (in the form of "forward delivery contracts") to pay dollars—for foreign funds when the dollar is strong against another currency, in the anticipation of buying back a greater number of those dollars (or "covering" the IOU's) when the foreign currencies strengthen against the dollar. Aside from possessing acumen in sensing the changing values of currencies against each other, the trader must be constantly aware of how much of each currency the bank maintains in its investment portfolio. SAS becomes an ideal programming tool for converting the cluttered assortment of data on currency trading contracts into a concise, readable transaction report. Thus traders are better kept aware of their currency positions when managing a bank's investment portfolio in foreign funds.

Concepts Involved:

Several fundamental methods for profiting in foreign currencies are available, namely:

1. Buying LONG,
2. HEDGING, and
3. Selling SHORT.

These methods, for a U. S. dollar based trader, can be defined as follows:

1. Buying LONG refers to trading dollars for a foreign currency (or for somebody else's promise to deliver a certain amount of a foreign currency by a certain date), and holding onto that currency (or promise) for trading back into dollars by a specified future date.

2. HEDGING refers to guarding against excessive trading losses by protecting LONG positions with SHORT sales contracts, and protecting SHORT sales contracts with LONG positions.

3. Selling SHORT refers to trading promises to deliver a foreign currency amount on a specified date, in exchange for dollars received now.

Several computer programming concepts in the Macro Language can be confusing unless specifically defined, namely:

1. PARAMETER -- a "value." Any word or number in an equation.
2. MACRO -- a subroutine, here coded in the SAS macro language.
3. "NEW" MACRO--a subroutine coded in the SAS macro language, as opposed to the "old" SAS macro style (which did not allow for passing of parameters or coding of conditional logic).
4. RESOLUTION -- translation of the parameters and code in a macro into meanings useful to a larger program.
5. CODE -- the values and statements in a computer program, i. e., the written lines of a computer program.
6. SYMBOLIC (or "substitution") PARAMETER -- a word or number preceded by an ampersand ("&"). The ampersand is the "symbol" which makes the word or number able to flexibly assume any actual value as assigned by the %LET or CALL SYMPUT functions in the SAS language.

Problem Statement:
HOW CAN A VARIETY OF FOREIGN CURRENCY TRANSACTIONS BE PROCESSED INTO ORGANIZED REPORTS ON DEMAND, WITH DYNAMIC AXES AND TITLE LINES REFLECTING THE TRADING FACTORS WHICH IMPINGE ON THE CONTENT OF EACH REPORT, TOGETHER WITH SPECIALIZED DATING AND VALUATION TECHNIQUES? The answer, of course, is to use the SAS System—but how can we make the SAS System do the job?

I. ALTERNATIVE SOLUTIONS USING SAS:

1. Replicate the roughly 200 lines of code needed to report on each currency, for the 16-20 currencies in the trading portfolio, or

2. Write the procedure only once, and teach the user how to change the subsetting-if statements in SAS as necessary to produce the desired reports, or

3. Write the procedure only once, including enough interactive messages to step the user through the procedure without his or her needing to know SAS, and using substitution parameters to automate the flow of the output without duplicating the code as the parameters change.

II. REVIEW OF ALTERNATIVES:

1. Replication of code to produce a 4000-line program is convenient in the sense that the user need only submit the job, walk away, and return in a while for the output. The code writer, furthermore, need not concern himself-or herself-with nesting of loops or calls to subroutines which might complicate the programming effort. Such a program can be wasteful in program storage space and computer time, however, and may often provide far more reports than the trader is interested in seeing on a given day.

2. Teaching the user to do his or her own program modifications is the Information Center manager's goal, and is certainly desirable from a support point of view. But not all users are willing to take the time to learn and maintain programming skills if their primary function is radically different from computer programming. And efficiency can be lost if a program must be modified and re-run for each desired report, since the trader must divide his concentration between modifying the program without making errors and executing currency trades without making errors.

3. Writing the procedure only once, and providing enough on-screen instruction at execution time to guide the casual user in running the program, offers maximum efficiency in both coding and use. Efficiency is increased through automation of the procedure, and potential for error is decreased since the need to change the coding of the program from report to report is alleviated. Coding of the program, with nested macro language calls and coordination of %LOCAL and %GLOBAL macro resolutions, however, is more difficult.

III. REVIEW OF SOLUTION: When one selects the third of the three alternatives, one will find that six DATA STEPS, six PROC STEPS, and four "new" macros can produce the desired reports, as follows:

1. DATA STEP # 1 reads in the raw data—in this case from a BMP conversion of an IMS database—and produces SAS datasets to form the basis of the graphed report. (The user of this logic can build his or her own DATA step as needed).

2. PROC STEP # 1, a PROC MEANS statement, provides a summation of the beginning balances for each foreign currency regardless of the number of different foreign bank accounts that are making the market in the foreign currency. (for example, up to five British banks share in the trading activity Manufacturer's Bank carries out in the British Pound Sterling. Balances in these five banks are summed to form the beginning balance of the trading day). An OUTPUT dataset of the beginning balances is produced for subsequent processing. Example follows:

(please see next page)
### DATA STEP # 2 Merges the Two SAS Datasets (from #1, above) to Relate Beginning Balances in Each Currency with the Transactions to be Applied Against that Balance. Example follows:

```sas
DATA FGN_BAL_FOII;  
IF CURRCODE="FOII" THEN FGN_BAL = FGN_BAL + PURCCH菅DFGN - SELDFGN;
ELSE DATADATE =.;
RUN;
```

### DATA STEP # 3 Subsets the OUTPUT Dataset as Mentioned Above into a Beginning Balance Dataset for Each Individual Currency, Complete with Exchange Rates in Effect at the Time Between that Currency and the U.S. Dollar. This Step Also Outputs a Separate SAS Dataset for Each Foreign Currency in the Portfolio. Example follows:

```sas
DATA FGN_BAL_FOII;  
IF CURRCODE="FOII" THEN FGN_BAL = FGN_BAL + PURCCH菅DFGN - SELDFGN;
ELSE DATADATE =.;
RUN;
```

### DATA STEP # 4 Merges the Two SAS Datasets (from #1, above) to Relate Beginning Balances in Each Currency with the Transactions to be Applied Against that Balance. Example follows:

```sas
DATA FGN_BAL_FOII;  
IF CURRCODE="FOII" THEN FGN_BAL = FGN_BAL + PURCCH菅DFGN - SELDFGN;
ELSE DATADATE =.;
RUN;
```

### DATA STEP # 5 is Embedded in the Macro %FILLINS, and Creates a Subset of the Previously MERGED Dataset Appropriate to the Resolution of %CURR (the Macro Substitution Parameter). This Subset Also Contains Contrived Observations Where Needed, to Complete the Twice-Monthly Periodicity Pattern Desired by the User. (Please Refer to Macro # 2 Code on Following Page for Example). Example follows:

```sas
%MACRO FILLINS;  
DATA FGN_BAL_FOII;  
IF CURRCODE="FOII" THEN FGN_BAL = FGN_BAL + PURCCH菅DFGN - SELDFGN;
ELSE DATADATE =.;
RUN;
%MEND FILLINS;
```

### DATA STEP # 6, Embedded in the Macro %FILLINS, is a NULL Dataset, and Serves to Pass the Values from the PROC MEANS Output Datasets (as Mentioned Above) to a TITLE Line, for Inclusion in the Report on Each Currency. The CALL SYMPUT Function is the Principal Technique in this DATA Step. Example follows:

```sas
%MACRO TITLES;  
DATA NULL;  
SET "FOII" IF CURRCODE="%CURR";  
TITLE1=TITLE="FOII",DATADATE,DATE2;  
CALL SYMPUT("CURRC",CURRCODE);  
CALL SYMPUT("SELLSPOT",SELLSPOT);  
CALL SYMPUT("BUY3D",BUYSPOT);  
CALL SYMPUT("BUY3U",BUY3D);  
CALL SYMPUT("BUY240",BUY240);  
CALL SYMPUT("BUY90",BUY90);  
CALL SYMPUT("BUY60",BUY60);  
CALL SYMPUT("BUY30",BUY30);  
CALL SYMPUT("BUY15",BUY15);  
CALL SYMPUT("BUY0",BUY0);  
CALL SYMPUT("DATADATE",DATE2);  
RUN;
%MEND TITLES;
```

### MACRO # 1, Named "%DRAWGRAP," is a Subroutine to Call Several Processing Options Applicable to All the Outputs. A %MACRO was not necessary for this activity, but is convenient. Example follows:

```sas
%MACRO DRAWGRAP;  
TSS FREE F(FT20FOOI);  
TSS AllOC FCFT20FOOI) DAlC224SAS.OUTUPJ  
OPTIONS CENTER DQUOTE TLS"132 PS"60J;  
%MEND DRAWGRAP;
```

### MACRO # 2, Named "%FILLINS," is a Subroutine for Organizing DATA Steps # 4 and 5, as Mentioned above, into Proper Sequence for Report Production. This Subroutine is the Best Point for Performing Any Data Step Manipulation Prior to Reporting. Example follows:

```sas
```
11. MACRO # 4, named "ZDRAW," is a subroutine for assigning the TITLE lines to the proper report output, and drawing the desired graph with PROC CHART. Example follows:

```
:%MACRO DRAW; 
%PUT DRAWING FOR &CURR PROCESSING NOH TITLEI POSITION CMART OF &CURR, AS OF "311 A. M. ON &SYSDATE; TITLEI USING DATA FROM CLOSE OF &DATADT; THE6 MOST ACTIVE CURRENCIES, IN LOCAL DENOMINATION; TITLE8 &CURRCI) SELL &SU.Si>T iSlUO iSlL6;;' "SU9U &SlLl:;~ J.::llHC; SKIP; TITLE8 NOTE: "PROFITI'tOSS IF CLOSED TODAY- INDICATED BELOW GRAPH; TITLE9 TODAYS FULLY HEDGED PROHT POTENTIAL IN ICURRCD IS $ &PROFlT; SKIP 3; PROC CHART; HBAR DATE ;' DISCRETE ;' ENDGU / END %MACRO; 
```

12. MACRO # 5, named "ZGRAPH," is the umbrella subroutine which coordinates the flow of the actual report production from currency to currency. Examples of the closing lines in and calls to ZGRAPH (along with a few lines to control output processing to TSO) follow:

```
:%MACRO CHART; 
%PUT OUT DISPLAY FOR &CURR PROCESSING NOH TITLEI POSITION CMART OF &CURR, AS OF "311 A. M. ON &SYSDATE; TITLEI USING DATA FROM CLOSE OF &DATADT; THE6 MOST ACTIVE CURRENCIES, IN LOCAL DENOMINATION; TITLE8 &CURRCI) SELL &SU.Si>T iSlUO iSlL6;;' "SU9U &SlLl:;~ J.::llHC; SKIP; TITLE8 NOTE: "PROFITI'tOSS IF CLOSED TODAY- INDICATED BELOW GRAPH; TITLE9 TODAYS FULLY HEDGED PROHT POTENTIAL IN ICURRCD IS $ &PROFlT; SKIP 3; PROC CHART; HBAR DATE ;' DISCRETE ;' ENDGU / END %MACRO; 
```

Several uniquely powerful functions of the SAS base product make this program even more completely automated:

I. The INPUT function, at invocation time, allows the execution of this application in a fashion completely transparent to the user.
(uniquely powerful features, cont.)

2. Automatic sizing of graph axes at execution time allows each graphed currency to be represented according to its own denomination—rather than being pegged to a fixed vertical axis template.

3. The $ option allows the procedure to be executed in batch regardless of the record size limitations that the operating system may place on other data base programming techniques using JCL.

4. The INCLUDE function, especially when used in conjunction with the $ options and its automatic influence on the setting of the S2 option, allows further economy of code. The same 200-line program which is executed interactively can, because of the $ and S2 options, be executed in batch with no modification! Example follows:

```
//STEP3 EXEC S5AS,EDIT=INSK,OPTIONS=\"MACRO DEQUOTE S=0 LABEL\"
//STEP4 EXEC S5AS,EDIT=INSK,OPTIONS=\"MACRO DEQUOTE S=0 LABEL\"
//STEP5 EXEC S5AS,EDIT=INSK,OPTIONS=\"MACRO DEQUOTE S=0 LABEL\"
//STEP6 EXEC S5AS,EDIT=INSK,OPTIONS=\"MACRO DEQUOTE S=0 LABEL\"
//STEP7 EXEC S5AS,EDIT=INSK,OPTIONS=\"MACRO DEQUOTE S=0 LABEL\"
//STEP8 EXEC S5AS,EDIT=INSK,OPTIONS=\"MACRO DEQUOTE S=0 LABEL\"
```

Example Output:

1. VBAR chart of Italian Lira portfolio balance and trading profitability.

2. HBAR chart of Italian Lira portfolio balance and trading profitability.

3. VBAR chart of German Mark portfolio balance and trading profitability.

4. HBAR chart of Japanese Yen portfolio balance and trading profitability.
Macro Programming Tips:

1. Always set the value of a symbolic parameter OUTSIDE of the macro in which that value is to be used, otherwise the macro in which the value is to be used will not resolve. In this example, the macro &TITLES sets the value of parameters to be used in the various charts--but the titles themselves are applied by the macro %DRAW. Both of these macros are nested in the larger macro, %GRAPH--but neither is nested within the other. Similarly the values for the changing currencies are set outside of all of the macros, and therefore become available to all of the macros.

2. The macro calling structure for this application is as follows:

```
%MACRO GRAPH;
  %PUT Scoring Processing;
  ;
  &TITLES;
  &FILLING;
  &DRAW;
  %MACRO DRAW;
  %LET CURR=DEM;
  %GRAPH;
  %LET CURR=GBP;
  %GRAPH;
  %LET CURR=CAD;
  %GRAPH;
  %LET CURR=CHF;
  %GRAPH;
  %LET CURR=FRF;
  %GRAPH;
%ENDGRAPH;
%ENDMACRO;
```

For further clarification or discussion the author may be contacted at the following address:

Blue Cross and Blue Shield of MI
600 E. Lafayette Dept J006
Detroit, MI 48226

Phone (313) 225-7913

SAS is a registered trade mark of the SAS Institute, Inc.

Conclusion:

The SAS macro language is shown to be especially valuable in situations where a variable number of financial reports are desired with a minimum of code writing and processing inefficiency. The use of symbolic variables to determine the flow of logic in subsetting-if statements allows economies of coding during selective processing activities.

Logic similar to that used in this SAS macro language application has been successfully used in subsequent projects to compare health care billings from health care providers against each other, and for preparing help screens to guide users through complicated SAS procedures such as PROC XII, and more.

The SAS macro language is shown to be especially valuable in situations where a variable number of financial reports are desired with a minimum of code writing and processing inefficiency. The use of symbolic variables to determine the flow of logic in subsetting-if statements allows economies of coding during selective processing activities.

Logic similar to that used in this SAS macro language application has been successfully used in subsequent projects to compare health care billings from health care providers against each other, and for preparing help screens to guide users through complicated SAS procedures such as PROC XII, and more.