How Symbolic Variables Can Reduce Code in a Graphics Environment

Monique Bryher / MRI Consulting, Inc., Los Angeles, CA

ABSTRACT

This paper discusses how to g enerate a series of 4-way bar charts that are replayed in SAS/GRAPH® when the list of possible values assigned to the independent variable is changeable. Macro language coding removes the necessity of maintaining a hard-coded list that might have to be modified during the reporting period, e.g., monthly, by having SAS assign each value of the independent variable to a symbolic variable, which is then resolved during the production of the graph. This eliminates the need for writing code to create each graph, resulting in a program that is shorter and easier to maintain. Last, examples of code will be given which illustrate the power of using double and triple ampersands derived from data variables to supply information required by graphics language expressions, such as names, labels, titles, and footnotes.

INTRODUCTION - A PROGRAM TO GENERATE 4-WAY GRAPHS FOR A UTILITY PLANT

The concepts presented in this paper will be illustrated by referring to a program that produces graphs on a monthly basis. The program graphically displays the number of open work orders for construction trades at a municipal utility plant by the following categories:

1) The current status of each work order (e.g., whether the work order is in the planning stage or has been scheduled).
2) The "age" or length of time since the work order became opened/active, expressed in number of days, which are divided into four (4) categories.
3) Data are displayed for the current month and the four prior months so that potential sources of backlogs can be detected.

The desired output will contain one (1) trade per page, with each of the four status code categories being represented by its own graph. For each status code/graph, the past five (5) months of work order activity will be displayed in vertical bar chart format, along with the number of days the work orders in that status category have been open (refer to the attached sample output).

There are three (3) basic techniques this program will address:

1) Build a table of names and titles which will be used in place of hard-coded values in the graphs. This will free us from creating a graph for each trade and status code. In the original application, this would have required up to 40 graphs to be coded; using symbolics reduces that number to 1.
2) Use of the CALL SYMPUT function to link the trade with its description, or title;
3) Use of multiple ampersands (&) and the period (.) symbol and their meaning in SAS macros;

STEP 1:

The first requirement is to establish a series of macro (or symbolic) variables which will describe the facility name, valid trades, status codes, and the age of work orders. Here is the code to set-up those symbolics:

```
*********************************;
* SET SYMBOLIC VARIABLES  ;
*********************************;
*;
DATA _NULL_;
 %let plant =  METROPOLIS POWER PLANT;
* ;
 %let msgca = CARPENTER;
 %let msgel  = ELECTRICAL;
 %let msgin  = INSTRUMENT;
 %let msgma = MACHINIST;
 %let msgme = MECHANIC;
 %let msgpl  = PLUMBER;
```

```
* ;
 %let stat1 = 1;
 %let stat2 = 2;
 %let stat3 = 3;
 %let stat4 = 4;
* ;
 %let msg1b = STATUS 20,30,40;
 %let msg2b = STATUS 50;
 %let msg3b = STATUS 60;
 %let msg4b = STATUS 70-80;
* ;
 %let msg1a = WORK IN PLANNING;
 %let msg2a = WAITING FOR MATERIALS;
 %let msg3a = WAITING TO BE SCHEDULED;
 %let msg4a = SCHEDULED OR IN PROGRESS;
 RUN;
```

STEP 2:

The second step - building a list of the trades to be used in the current run - requires reading the data set and using the FIRST. feature found in Base SAS to select each trade:

```
*****************************;
* CREATE LIST OF TRADES  ;
*****************************;
* ;
DATA TRADELST(KEEP = TRADE);
 SET NEWDET;
 BY TRADE;
 IF FIRST.TRADE;
 RUN;
```

Next, we need the trades arranged so that a loop can be established to create one graph for each trade when the macro is run. The list should look something like this (if there were data on all of the trades):

```
TRADE1 = CARPENTER
TRADE2 = ELECTRICAL
   . . .
TRADEn = PLUMBER
```

where "n" is the last trade found on the data set. In our example, only one (1) trade existed on the data set for the month we are illustrating; hence, we will have only

```
TRADE1 = TRADEn = ELECTRICAL
```

In order for us to be able to access the trade assignments anywhere in the program, we will need to place them in a symbolic variable by using the CALL SYMPUT function:

```
TRADENAM =
   'TRADE'||TRIM(LEFT(PUT(CTR,2.)));
CALL SYMPUT(TRADENAM,TRADE);
```

In our example, the name TRADE1 will be placed in the variable TRADENAM. The CALL SYMPUT function will then place the name of the current trade (in this case 'EL' for ELECTRICAL) in a macro variable called TRADE1. If there were more trades in our data set, e.g., 'IN' then we would have a variable TRADE2 = 'IN', etc.

STEP 3: The Driver Macro

The next step in our project is to write a macro which will generate or "drive" the generic chart macro which we will assemble in STEP 4. This macro will need to print four (4) charts for each trade, each one representing a status code category. The total number of trades is contained in the &maxctr symbolic variable.

```
*************************************;
* macro to drive charts and replay macros ;
*************************************;
* ;
%macro driver;
 %do ctr = 1 %to &maxctr;  ← # trades
   %do num = 1 %to 4;  ← 4 status codes
     %charts;  ← our generic bar chart
   %end;
%replay;  ← replay each 4-way chart
%end;
%mend driver;
```

STEP 4: The Chart Macro

The %chart macro illustrates powerfully the amount of code that can be reduced by exploiting the Macro Facility:

```
*****************************;
* macro to create charts by trade ;
*****************************;
```

```
* ;
%macro charts;
 GOPTIONS RESET    = GOPTIONS
                   DEVICE  = WINPRTC
                   ROTATE = LANDSCAPE
                   HSIZE     = 4 IN
                   VSIZE     = 5.25 IN
                   BORDER
                   NODISPLAY
                   NOCHARACTERS
                   ;
 TITLE1 H=.25;
 TITLE2 H=1.7 "&&msg&num.a";
 TITLE3 H=1.5 'AGE ANALYSIS';   ❶
 TITLE4 H=1.0 "&&msg&num.b";
  FOOTNOTE1  j=r  "CHART&num&&trade&ctr";
❷
 FOOTNOTE3 H=.15 ' ';
 LEGEND ACROSS=2 LABEL=NONE;
 PATTERN1 COLOR=BLACK VALUE=E;
 PATTERN2 COLOR=BLACK VALUE=L1;
 PATTERN3 COLOR=BLACK VALUE=R1;
 PATTERN4 COLOR=BLACK VALUE=X1;
 AXIS1 ORDER=(1 TO 5)
        VALUE=
    ❸ ("&dtl1" "&dtl2" "&dtl3" "&dtl4"  "&dtl5)
        label=(' ');
 AXIS2 LABEL=('# STEPS');
PROC CHARTDATA=NEWDET
         GOUT           =TRENDA;
  FORMAT AGECODE $AGEFMT.;
  WHERE TRADE    = "&&trade&ctr" and  ❹
              STATUS  = "&&stat&num";
  VBAR RUNCTR /
   ❺  NAME        = "CHART&num&&trade&ctr"
      SUMVAR     = WO_CTR
      SUBGROUP = AGECODE
      MAXIS        = AXIS1;
      RAXIS        = AXIS2
      LEGEND     = LEGEND
      WIDTH        = 7
      ;
  RUN;
%mend charts;
```

❶ Controls the description of the status code for each quadrant/graph and the codes themselves.

❷ Each quadrant/graph is assigned a unique name, which is posted to the bottom right.

❸ Macro variables for the most recent five (5) time periods.

❹ WHERE clause to indicate the trade and status code for that quadrant/graph.

❺ Name of the catalog entry (same as ❷ above).

STEP 5: The Replay Macro

```
**********************************;
* macro to replay charts in a template ;
*  -- by trade                          ;
**********************************;
* ;
%macro replay;
GOPTIONS HSIZE=0 VSIZE=0;
PROC GSLIDE GOUT=TRENDA
 NAME="CHART5&&trade&ctr";  ❻
 TITLE1  H=1.5
   'WORK ORDER BACKLOG AGE ANALYSIS
                          AS OF ' "&SYSDATE";
 TITLE2  H=1.5
  'TRADE - ' "&&&msg&&trade&ctr";  ❼
FOOTNOTE1 h=1.5 "&plant";
FOOTNOTE2 h=.8 ' ';
GOPTIONS DEVICE              = WIN
                TARGETDEVICE = WINPRTC
                RESET              = TITLE
                BORDER
                DISPLAY
                NOCHARACTERS
                ;
PROC GREPLAY IGOUT = TRENDA
           GOUT               = TRENDB
           TC                   = TEMPCAT
           NOFS;
 TDEF FIVEWAY
 1/LLX=6    LLY=48
   ULX=6    ULY=89
   URX=47  URY=89
   LRX=47  LRY=48

 2/LLX=53  LLY=48
   ULX=53  ULY=89
   URX=94  URY=89
   LRX=94  LRY=48

 3/LLX=6   LLY=7
   ULX=6   ULY=48
   URX=47  URY=48
   LRX=47  LRY=7
```

```
 4/LLX=53  LLY=7
  ULX=53  ULY=48
  URX=94  URY=48
  LRX=94  LRY=7

 5/LLX=0   LLY=0
  ULX=0   ULY=100
  URX=100 URY=100
  LRX=100 LRY=0
 ;
TEMPLATE FIVEWAY;
TREPLAY   1:"CHART1&&trade&ctr"
          2:"CHART2&&trade&ctr"
          3:"CHART3&&trade&ctr"  ❽
          4:"CHART4&&trade&ctr"
          5:"CHART5&&trade&ctr"
           ;
 RUN;
QUIT;
%mend replay;
```

❻ Name of the GSLIDE chart catalog entry.
❼ Description of the trade, e.g., CARPENTER
❽ Name of the catalog entry for the four charts and the GSLIDE chart.

STEP 6: Running the Program

Now for the fun part - let's see how the symbolic variables resolve when the macro DRIVER is started:

1) The variables CTR and NUM are both set to 1 and the %chart macro is executed.
2) The TITLE2 statement is translated as follows:

```
          ①              ②
&&msg&num.a → &msg1a →
                    "WORK IN PLANNING"
```

① During the first stage of translation, the macro compiler resolves &num to the number 1 and removes the first & from &&msg. The text resolves to ② &msg1a. This is a recognizable symbolic, which is finally translated to the literal "WORK IN PLANNING".

Note the use of the period ('.') in the text string; it's use is to indicate the end of a macro variable name to the compiler. In this case, there is no macro variable called &numa, but there is one called &num.

3) The TITLE4 statement is similarly translated:

```
&&msg&num.b → &msg1b →
                    "STATUS 20, 30, 40"
```

4) The FOOTNOTE1 statement gives a unique name to the quadrant/chart. This is how it resolves:

```
CHART&num&&trade&ctr → CHART1&trade1
                     → CHART1EL
```

5) The macro variables &dtl1 through &dtl5 were set using CALL SYMPUT statements earlier in the program, where &dtl1 is the oldest reporting period and &dtl5 is the most recent.

6) The WHERE statement determines the selection of data by trade and status code. The status code determines which quadrant/chart is being produced, e.g., upper left. These are the steps in the translation process:

```
&&trade&ctr → &trade1 = CARPENTER
&&stat&num → &stat1   = 1 (status 20, 30, 40)
```

7) Finally, a catalogue entry name that is identical to the footnote at the lower right of each chart is created.

The above seven (7) steps will occur four (4) times for each trade, assuming that there is activity on all status codes.

After the charts for the first trade are created, the %replay macro is entered. The first portion utilizes PROC GSLIDE to draw a frame around the four charts and place the following super-titles over them, the second of which contains a 3-level deep translation:

```
TITLE2 H=1.0
 "TRADE - &&&msg&&trade&ctr"; →
 "TRADE - &&msg&trade1"    →
 "TRADE - &msgel"          → ELECTRICAL
```

The remainder of the macro variables from the PROC GSLIDE and PROC GREPLAY are easily resolved according to the above examples.

CONCLUSION

This paper has been a brief look at the benefits of writing macros and creating symbolic variables. By using the techniques described, programmers will reap a number of benefits:

1. Greatly reduce the size of the code by combining repetitive code into macros and using symbolic variables. Less code means less clutter too.
2. Concomittantly decrease syntax and typing errors by changing symbolic variable values defined in a list or table instead of having, as in the original application, up to 40 separate PROC GCHART paragraphs.
3. Standardization due to having to come up with a common routine.

References

SAS® Guide to Macro Processing,
Version 6, 2nd Edition
SAS® Macro Facility: Tips and Techniques,
Version 6, 1st Edition
SAS®/Graph Software, Volumes 1 and 2, Reference
Version 6, 1st Edition
The How-To Book for SAS/GRAPH® Software,
by Thomas Miron, 1995

Acknowledgements

Belated thanks to Larry Landers for having introduced me to some of the concepts utilized in this paper.

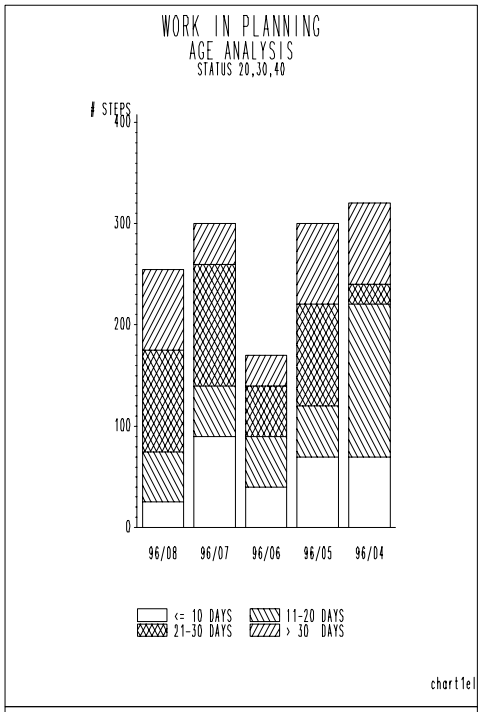
Author's Address

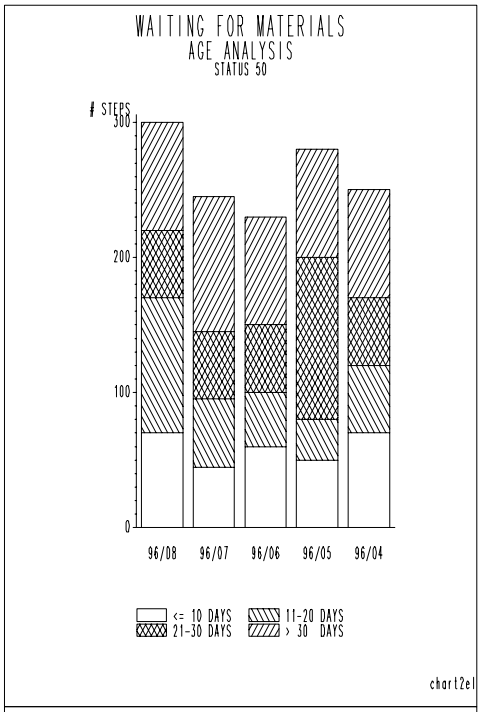
Monique Bryher, MSPH
MRI Consulting, Inc.
6043 Shirley Avenue
Tarzana, CA 91356
(818) 774-0043

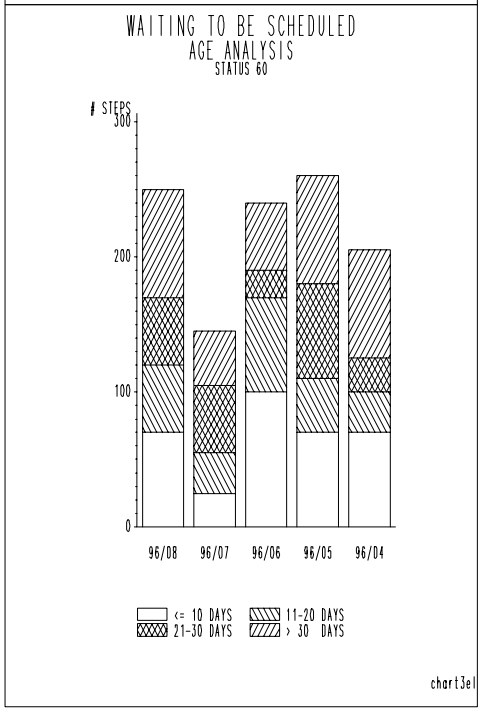
INTERNET: rebwest@aol.com

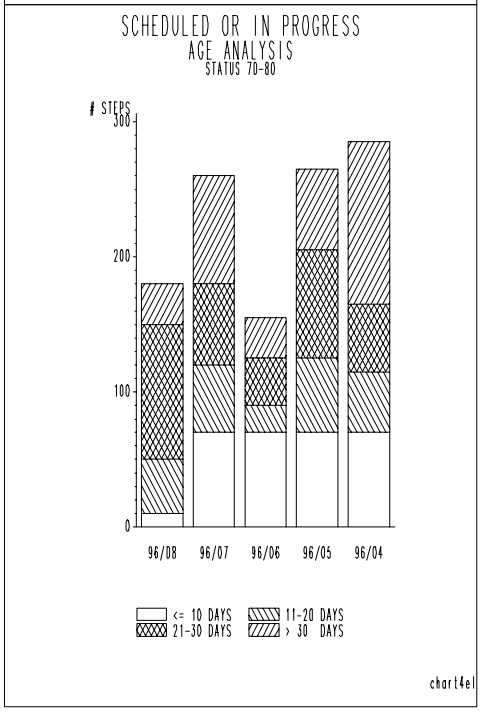
Please feel free to send your comments, suggestions, and experiences.

WORK ORDER BACKLOG AGE ANALYSIS AS OF 27AUG96
TRADE - ELECTRICAL



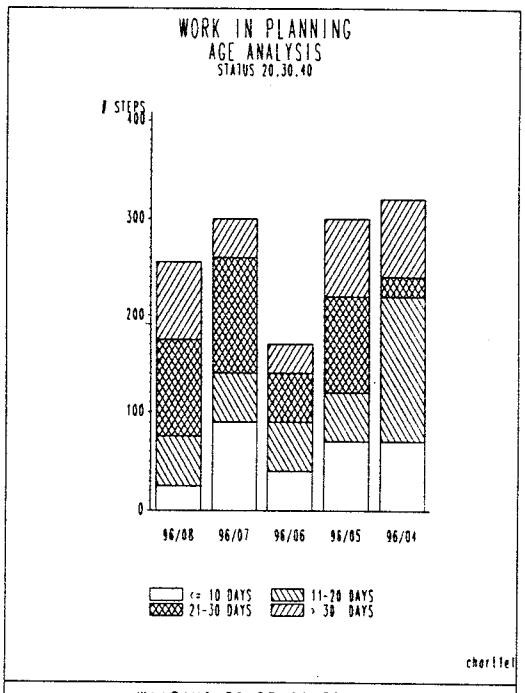


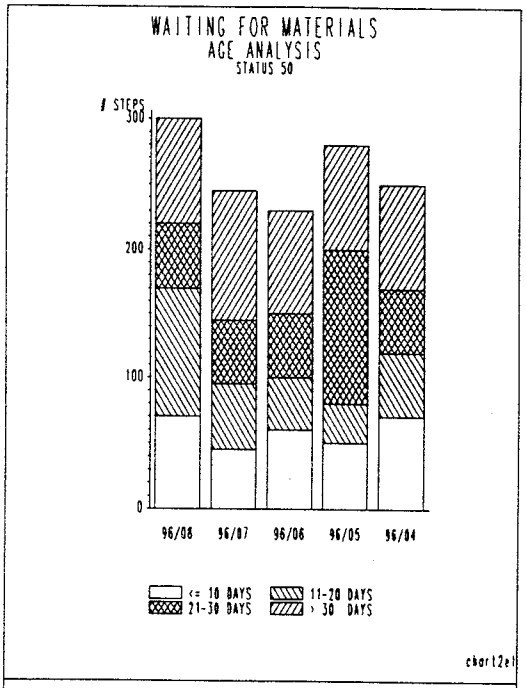


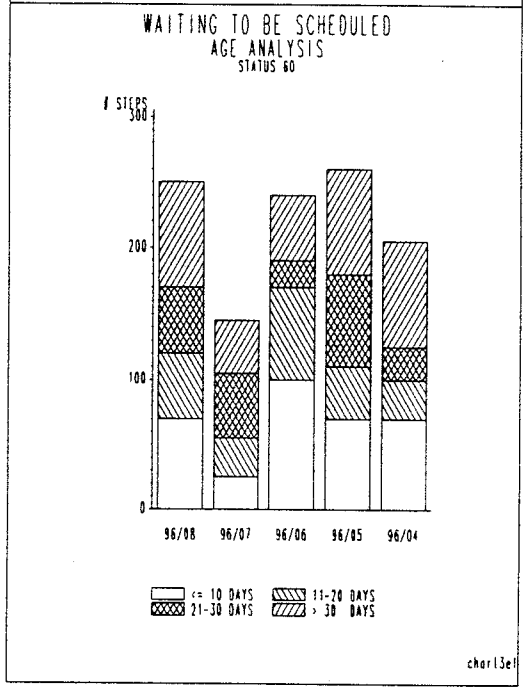


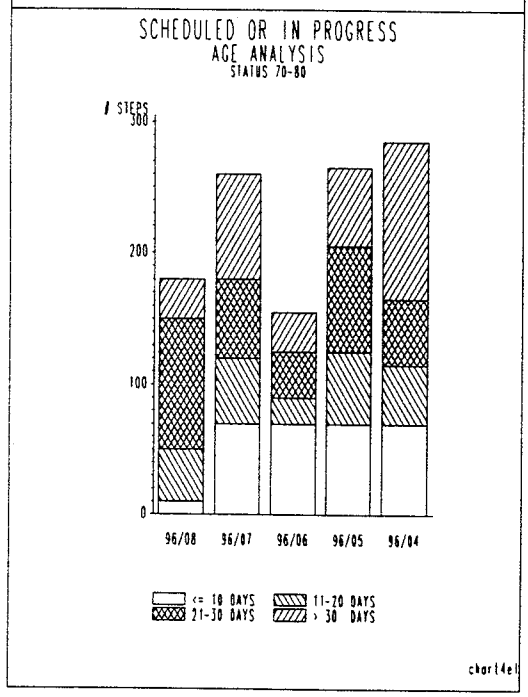
METROPOLIS POWER PLANT

WORK ORDER BACKLOG AGE ANALYSIS AS OF 27AUG96
TRADE - ELECTRICAL









METROPOLIS POWER PLANT