Generic SAS® Programs: User Control Using REXX

Arthur L. Carpenter
California Occidental Consultants

ABSTRACT

It is not unusual for several slightly different SAS programs to be written which do similar analyses. Often a prototype program is developed and then modified over the course of time into a series of programs. Perhaps all the programs are the same except for different input data bases or analysis variables. The maintenance of parallel programs such as these is very difficult. If a change must be made in one, then it must be made in all versions of the program. It is much easier to write and maintain one generalized program that will satisfy all of the users' needs. There are several methods of either converting SAS programs into boiler-plate or generic programs that can be used for multiple purposes. SAS macros and SAS AF are used extensively for this purpose. There is an additional need for non-SAS (yes, Virginia, there is non-SAS software) to interact directly with SAS programs.

An EXEC has been written in REXX which will convert generic SAS programs containing dummy arguments into executable SAS code. The user retains the flexibility of specifying which arguments are to be changed from within a calling EXEC.

The generic SAS program is written in standard SAS with the exception of the inclusion of dummy arguments which are replaced by the controlling EXECutive program prior to execution. As many dummy arguments as needed may be included in any one program. Because substitution of the arguments is done prior to execution, the passed strings can include quotations, special characters and may be used to selectively remove lines of code.

This paper discusses a simple example of a generic program and the advantages of using this type of controlling EXEC.

INTRODUCTION

The substitution of dummy arguments in SAS programs prior to the execution of the SAS code opens a wide range of possibilities for the user, as well as adding a flexible tool to the programmer's tool box. Generic programs by definition must be able to serve a variety of user's needs, while maintaining the ability to be easily used and understood. A simple SAS program to look at a data base is shown below:

```
PROC CONTENTS DATA=DBFISH.TRAWL;
PROC PRINT DATA=DBFISH.TRAWL (OBS=20);
```

This two line program does a PROC CONTENTS on a data base and then prints the first 20 observations using PROC PRINT. This is a trivial program, however, even it would be a useful tool if it could easily work on any data base. A REXX exec was written (for both CMS and PC-DOS) which allows a master or generic program to include dummy arguments, such as data base names, which can be replaced automatically prior to execution of the SAS program.

PROGRAM DESCRIPTION

A program written in REXX is used to control the conversion of master or generic SAS programs to executable SAS code. SASMaster or SASM's are SAS programs with dummy arguments. The SASMaster is converted into executable SAS code by substituting the dummy arguments, which are of the form *ARGi*, with the items or characters that are supplied by the user. Prior to execution the controlling EXEC edits the SASM, replaces the dummy arguments with the desired values and then executes the resulting SAS code. The above code would be more general if the user could specify the data set member of interest. This can be done by rewriting it as:

```
PROC CONTENTS DATA=DBFISH.*ARG1*;
PROC PRINT DATA=DBFISH.*ARG1* (OBS=20);
```

For this example the dummy argument *ARG1* is replaced by the data base member, TRAWL.

Any number of dummy arguments may be included in the SASMaster. In the above example it might be useful to specify both levels of the data base name:

```
PROC CONTENTS DATA=*ARG2*.*ARG1*;
PROC PRINT DATA=*ARG2*.*ARG1* (OBS=20);
```

One mark of a well written master program is the number of options available to the user. However, since not all options will be appreciated, the controlling exec must take into account unused options. In the above data base example, a third option could be in-

cluded to allow the user to specify the number of observations to print:

```
PROC CONTENTS DATA=*ARG2*.*ARG1*;
PROC PRINT
     DATA=*ARG2*.*ARG1* (OBS=*ARG3*);
```

As written, if the user fails to specify the number of observations, *ARG3* will not be resolved and the program will fail with a syntax error. One way to get around this is to eliminate all physical SASM records with unresolved dummy arguments. If the code is written with this in mind, the resulting SAS program will not only execute but can include program defaults:

```
PROC CONTENTS DATA=*ARG2*.*ARG1*;
PROC PRINT DATA=*ARG2*.*ARG1*
(OBS=*ARG3*)
;
```

If this code is executed without specifying the number of observations, then the entire line containing *ARG3* is deleted and the executable code becomes:

```
PROC CONTENTS DATA=DBFISH.TRAWL;
PROC PRINT DATA=DBFISH.TRAWL
;
```

The entire data set is printed as a default.

The removal of lines of code can be very useful. The arguments passed can depend on information presented by the user or as a result of operations performed by a controlling EXEC. In the following code a macro will be executed conditionally depending on the elimination of the line containing *ARG1*:

```
* *ARG1*
 %MRGUPD8(*ARG3*) RUN;
```

The call to macro %MRGUPD8 will be part of the comment starting on the previous line whenever *ARG1* receives a value e.g. COMMENT. This would result in:

```
* COMMENT
 %MRGUPD8(*ARG3*) RUN;
```

Since lines containing unresolved dummy arguments are deleted after the substitution, passing *ARG1* a value such as *ARG99* would cause the line to be deleted and the macro would execute.

SUMMARY

A REXX EXEC has been written which provides a flexible interface for users faced with the task of writing generic or master SAS programs. SAS programmers can write generic code containing dummy arguments that can be substituted into executable code by the controlling EXEC.

```
/*-------------------------------------------------------------------*/
/* SASM  --  convert to SAS jobs 08/18/88                            */
/*           MS/DOS version                                          */
/*                                                                   */
/* Used to create and execute a SAS (.SAS)  program from a SAS       */
/* master (.MAS).  This is done by substituting the second thru      */
/* nth arguments for the *ARG1* thru *ARGn-1* strings in the master  */
/* file using SASM.REX.  The first argument passed to this EXEC      */
/* is the name of the master file.                                   */
/* After the substitutions are made, the SASM master file            */
/* is renamed <sasname>.SAS in preparation for running SAS.          */
/*                                                                   */
/* Written by: Arthur L. Carpenter & J. Marandola                    */
/*             California Occidental Consultants                     */
/*             4239 Serena Avenue                                    */
/*             Oceanside, CA 92056                                   */
/*                                                                   */
/*             (619) 724-8579                                        */
/*                                                                   */
/* SAS is a registered trademark of SAS Institute Inc., Cary, NC.    */
/*-------------------------------------------------------------------*/
/* trace results */

options newcom

parse arg sasname arguments
if sasname = ? then call exp
```

```
if sasname = '' then do
        say 'ERROR: You must supply the SAS master name.'
        say 'Enter rx sasm ? for an explanation.'
        end
else do
     call cntargs
     call chkfile
     call edfile
     call removeargs
     call runSAS
     end
call exit


/*-------------------------------------------------------------------------*/
/* count the number of passed arguments                                    */
/*-------------------------------------------------------------------------*/
cntargs:
nargs = words(arguments)
if nargs >= 1 then do i = 1 to nargs
     a.i = word(arguments,i)
end
return

/*-------------------------------------------------------------------------*/
/* check to see that the SAS master exists                                 */
/*-------------------------------------------------------------------------*/
chkfile:
if dosdir(sasname||'.MAS') = '' then do
     say ' SAS master 'sasname'.MAS NOT FOUND'
     exit -2
end
return

/*-------------------------------------------------------------------------*/
/* edit the master create a SAS program file (.SAS)                        */
/*-------------------------------------------------------------------------*/
edfile:
if nargs > 0 then do
     queue top
     do i = 1 to nargs
          queue 'c'*arg'||i||'*''||a.i||'' * * '
     end
     queue set fn sasname
     queue set ft prg
     queue ffile
     kedit sasname||'.MAS (nomsg noscreen'
     if rc ^= 0 then exit rc
end
else copy sasname||.mas sasname||.prg
return

/*-------------------------------------------------------------------------*/
/* REMOVEARGS removes from .SAS file any lines still containing            */
/* *ARGx*.                                                                 */
/*-------------------------------------------------------------------------*/
REMOVEARGS:
queue 'top  '
queue 'done = 0'
queue 'do until done '
queue '     1/*ARG'
queue '     if rc ^= 0 then done = 1 '
queue '     if ^done then do          '
queue '          del'
queue '          ul'
queue '     end       '
queue 'end'
queue 'file'
kedit sasname||'.SAS (nomsg noscreen '
if rc ^= 0 then exit rc
return
```

```
/*-------------------------------------------------------------------*/
/* RUNSAS executes the new SAS program.                              */
/*-------------------------------------------------------------------*/
runSAS:

say '------ SAS ' sasname 'Passing arguments' arguments


SAS sasname||'.SAS'
code = rc
erase sasname||'.SAS'

if code ^= 0 then do
                  say 'SAS error.  See the templog. '
                  exit code
                  end

return

/*-------------------------------------------------------------------*/
/* exit the program.                                                 */
/*-------------------------------------------------------------------*/
exit:
exit
return

/*-------------------------------------------------------------------*/
/* EXP provides an explanation of how SASM works                     */
/*-------------------------------------------------------------------*/
EXP  :

clrscrn

say 'SASM is used to pass arguments at execution time to SAS          '
say 'programs (.SAS).                                                 '
say 'The SAS master to be executed must have an extension of .MAS .   '
say '                                                                 '
say 'SASM executes SAS for the file <fn>.SAS using the  arguments     '
say '<arg1>, <arg2> to <argn>.  <fn>.SAS is built from the file       '
say '<fn>.MAS using this REXX EXEC.                                   '
say 'The arguments <arg1>, <arg2>, to <argn> replace the strings      '
say '*ARG1*, *ARG2* to *ARGn* in the .MAS file.                       '
say 'The lines containing all extra *ARG* arguments are deleted.      '
say '                                                                 '
say 'Format for execution:                                            '
say '   rx sasm <filename> <arg1> <arg2>.......<argn>                  '
say '          where arg1 to argn are optional                        '
say '          and the master file must have an extension of .MAS  .  '

exit
return
```

Arthur L. Carpenter
CALOXY
4239 Serena Avenue
Oceanside, CA 92056

(619) 724-8579