

CHAPTER

9

The SAS Interface to REXX

<i>Overview</i>	83
<i>The Subcommand Environment</i>	84
<i>Retrieving and Assigning the Values of REXX Variables in a SAS Program</i>	85
<i>Using the GETEXEC DATA Step Function</i>	85
<i>Using the PUTEXEC Call Routine</i>	85
<i>Routing Messages to the SAS Log</i>	86
<i>Return Codes from SAS Statements Submitted by a SASMACRO</i>	87
<i>RC Variable</i>	87
<i>RC Values</i>	87
<i>Return Codes from SASMACROs</i>	88
<i>Mode Switching</i>	88
<i>Productivity Aids for Interactive SAS Sessions on CMS</i>	89
<i>Invoking a SASMACRO from the Windowing Environment Command Line</i>	90

Overview

REXX is a general purpose, high-level procedural language that is well known for combining powerful programming features with ease of use. REXX is also well known for its strengths as a macro language, most commonly with XEDIT. The SAS REXX macro facility enables you to use REXX as a macro language with SAS. This expands the programming possibilities available to you with SAS under CMS.

The REXX macro facility can be used to do these things that the SAS macro facility can do:

- dynamically create and submit SAS statements
- communicate information between SAS steps
- route messages to the SAS log
- issue commands to CMS
- pass information between SAS and CMS using GLOBALV variables.

You can also use the REXX macro facility to

- pass information between SAS and CMS using REXX variables
- temporarily transfer control from an executing macro back to an interactive SAS session
- create pseudo SAS windowing environment commands
- transfer control to a CMS application (for example, XEDIT or FILELIST), and invoke SAS commands from within them.

To use the REXX macro facility, write a REXX macro program and submit it from within a SAS session. A SAS REXX macro is a REXX program with a filetype of

SASMACRO and a filename that you choose. A SASMACRO can contain SAS statements in addition to REXX code. When a SASMACRO is running, any instruction that REXX interprets as an external command is submitted to SAS for execution. For example, the following program, TRYIT SASMACRO (a version of which is shipped with SAS), submits a DATA step and a PROC step to SAS:

```

/* TRYIT SASMACRO - a SAS REXX macro          */
/* process arguments, establish defaults      */
parse arg dname argname .
if dname = '' then dname = 'a'
if argname = '' then argname = 'x'

/* Pass a DATA step and PROC step into SAS */
'data' dname';' argname'=1; run;'
'proc contents; run;'

exit      /* Return to SAS                    */

```

To run a SASMACRO, submit its CMS filename and any arguments as if it were a SAS statement. This effectively makes the SASMACRO an extension of the SAS language. A statement invoking a SASMACRO may be placed anywhere in a SAS program. When SAS encounters a statement that it does not recognize, it passes the statement to the SAS REXX macro facility for execution. If there is no SASMACRO by the specified name, SAS flags the statement as invalid and prints an error message in the log. A SASMACRO cannot have a name that is the same as a valid SAS statement or its abbreviation. To invoke the TRYIT SASMACRO example with two arguments, submit the following statement to SAS:

```
TRYIT A B;
```

This creates SAS data set A containing one observation with one variable B, which has a value of 1.

A SASMACRO is invoked when SAS encounters it, just as any other global SAS statement in that location would be. A SASMACRO invoked within a DATA step is executed only one time, when the DATA step is compiled. It is not executed for each observation in the DATA step. Any SAS statements submitted from the SASMACRO are processed before subsequent statements are processed in the invoking SAS program.

The Subcommand Environment

When a REXX SASMACRO receives control, the default subcommand environment is 'SAS'. This causes REXX to pass external commands to SAS. You can use the REXX ADDRESS instruction, followed by the name of an environment, to change to a different default subcommand environment. For example:

```

address command
address xedit
address sas

```

A SASMACRO is analogous to an XEDIT macro in its structure and invocation:

- A SASMACRO is a REXX program. It is submitted as part of a SAS program in the same way as any other global SAS statement. A SASMACRO submits SAS statements through the SAS subcommand environment by specifying or defaulting to 'SAS' as its "address."

- An XEDIT macro is a REXX program. It is executed from the XEDIT command line in the same way as any other XEDIT subcommand. An XEDIT macro submits XEDIT subcommands through the XEDIT subcommand environment by specifying or defaulting to 'XEDIT' as its "address."

From the time a SASMACRO is invoked until it is terminated, the SAS subcommand environment is available. For example, a SASMACRO can invoke a REXX exec which issues the following instruction and submits SAS statements:

```
address sas
```

You can use a REXX exec in this way to layer and reuse the code in a SASMACRO.

Although SASMACROs and EXECs can be nested, the SAS subcommand environment itself is not re-entrant; that is, it cannot accept a new SAS statement while a previous statement that is submitted via ADDRESS SAS is still active. The limitation that this imposes is that a SASMACRO cannot use the X statement or the CMS statement to execute a command that, in turn, issues SAS statements via ADDRESS SAS. Instead, use ADDRESS command or ADDRESS CMS to execute such a command.

Each command string submitted to the SAS subcommand environment is limited to a maximum length of 132 characters. This limit is on the final length of the string which results from the completion of all interpretation by REXX.

Retrieving and Assigning the Values of REXX Variables in a SAS Program

Using the GETEXEC DATA Step Function

During the execution of a SASMACRO you can retrieve the value of any exposed REXX variable from the current REXX program into a DATA step by using the GETEXEC function. The GETEXEC function is analogous to the SYMGET function in the SAS macro facility.

The following SASMACRO submits a DATA step that uses the GETEXEC function to retrieve the value of the REXX variable DATALINE. It then prints the value of DATALINE in the SAS log. Note that the REXX variable name must be in uppercase.

```
/* GETLINE SASMACRO - a SAS REXX macro      */
dataline='This data will be placed into the SAS',
        'data set A'
/* Pass a DATA step to SAS                */
"data a;"
"x=getexec('DATALINE');"
"put x;"
"run;"
exit
```

For more information about the syntax and usage of GETEXEC, see "GETEXEC" on page 141.

Using the PUTEXEC Call Routine

During the execution of a SASMACRO, you can assign a value to a REXX variable in the current REXX program from a DATA step by using the PUTEXEC call routine. The PUTEXEC call routine is analogous to the SYMPUT routine in the SAS macro facility.

The following SASMACRO submits a DATA step that assigns the value of a SAS variable to the REXX variable SASTIME. The SASMACRO then displays the value of the REXX variable on the console. Note that the REXX variable name must be in uppercase.

```
/* PUTTIME SASMACRO - a SAS REXX macro      */
"data _null_;"
  "time=symget('SYSTIME');"
  "call PUTEXEC('SASTIME',time);"
"run;"
say 'SAStime is' sastime
exit
```

For more information about the syntax and usage of PUTEXEC, see "CALL PUTEXEC" on page 124.

Routing Messages to the SAS Log

A set of directives is available to SASMACRO programs to control printing to the SAS log. SASMACRO directives use a leading ++ sequence to differentiate them from normal SAS language statements. The directives must be in uppercase and cannot end in a semicolon. They must also be submitted on a separate line from other SAS statements.

Three directives are available for controlling printing to the SAS log:

'++SASLOG'

causes subsequent SAS statements submitted from the SASMACRO to be printed in the SAS log.

'++NOLOG'

causes subsequent SAS statements submitted from the SASMACRO not to be printed in the SAS log. This is the default setting.

' ++SASLOG *message-text*

prints *message-text* in the SAS log and causes subsequent SAS statements submitted from the SASMACRO to be printed in the SAS log.

The following example uses the ++SASLOG directive to place text and subsequent SAS statements in the SAS log:

```
/* An extended version of TRYIT SASMACRO      */
/* process arguments, establish defaults      */
parse arg dname argname .
if dname = '' then dname = 'a'
if argname = '' then argname = 'x'

/* Place text into SAS log                    */
/* and subsequent SAS statements              */
'++SASLOG Running the TRYIT macro.'

/* Pass a DATA step and PROC step into SAS */
'data' dname';' argname'=1; run;'
'proc contents; run;'

exit /* Return to SAS                        */
```

Output 9.1 on page 87 shows the resulting SAS log:

Output 9.1 SAS Log for TRYIT SASMACRO

```

1      tryit;
++++  Running the TRYIT macro.
++++  data a; x=1; run;
NOTE: The data set WORK.A has 1 observations and 1 variables.
++++  proc contents; run;

```

Return Codes from SAS Statements Submitted by a SASMACRO

RC Variable

In a REXX program, the special variable RC is always set when any command string is submitted to an external environment. Ordinary execs submit CMS commands. When the CMS command completes and control is returned to REXX, the RC variable is set to the return code from the CMS command.

The RC variable is set in a slightly different way for a SASMACRO. The strings that are submitted to SAS are not necessarily complete execution units. SAS collects SAS language elements until it encounters a RUN statement, at which point it runs the SAS step. The RC variable is set to 0 when partial program fragments are submitted. The SAS return code is assigned to the REXX variable RC only for the string that contains the RUN statement.

RC Values

The value of the REXX RC variable is set to the value of the &SYSERR automatic SAS macro variable in all but four cases:

- when an attempt is made to enter the SAS subcommand environment recursively. In this case, the statement is ignored and the RC value is set to -2.
- when a SASMACRO is active and the SAS session is terminated with a BYE command, or with an ENDSAS command or statement. In this case, the SAS session returns control to the SASMACRO. SAS statements that are subsequently submitted by the SASMACRO cannot be executed, and the RC value is set to -3 for each.
- when a SASMACRO is active and SAS is interrupted with an attention and the SAS task is cancelled. In this case SAS statements that are subsequently submitted by the SASMACRO are not executed, and the RC value is set to -4 for each.
- when an attempt is made to submit a command string that is longer than 132 characters. In this case, the statement is ignored and the RC value is set to -5. An error message and the first 132 characters of the string are written to the SASLOG.

The following RCTEST SASMACRO demonstrates when the REXX variable RC gets set:

```

/* RCTEST SASMACRO - a SAS REXX macro */
/* Show SAS statements in the log */
'++SASLOG'
'data x;'

```

```

'do i = 1 to 10;'
'output;'
/* show rc in the SAS log */
'++SASLOG The RC value is:' rc
'run;'
/* show rc in the SAS log */
'++SASLOG The RC value is:' rc
exit

```

Output 9.2 on page 88 shows the resulting SAS log.

Output 9.2 SAS Log for RCTEST

```

1  rctest;
++++ data x;
++++ do i = 1 to 10;
++++ output;
++++ The RC value is: 0
++++ run;

5  run;
   --
   117
ERROR 117-185:  There were 1 unclosed DO blocks.

NOTE:  The SAS System stopped processing this step because of errors.
WARNING:  The data set WORK.X may be incomplete.  When this step was stopped,
          there were 0 observations and 1 variables.
++++ The RC value is: 1012

```

Return Codes from SASMACROs

The return code from a SASMACRO is not automatically available to SAS. To pass information like a return code from a SASMACRO back to SAS, you can use the GETEXEC DATA step function described in “Using the GETEXEC DATA Step Function” on page 85.

Mode Switching

In addition to being able to submit SAS language statements from a SASMACRO, you can use a SASMACRO to temporarily pass interactive control to the SAS session. This environment, called SAS SUBSET, is similar in concept to CMS SUBSET. A SASMACRO (or any REXX program executed while a SASMACRO is active that has ADDRESS SAS in effect), can pass control to SAS SUBSET by issuing the following directive:

```
'++SAS'
```

To exit SAS SUBSET and return to the suspended exec, submit the following SAS statement:

```
cmsreturn;
```

CMSRETURN is accepted as a valid SAS statement only when SAS SUBSET is in effect, and it cannot be submitted from a SASMACRO.

The following example demonstrates switching from a SAS session to an XEDIT session to SAS SUBSET and back again using a SASMACRO and an XEDIT macro.

The program named SASXEDIT SASMACRO shows how to temporarily pass control to XEDIT:

```
/* SASXEDIT SASMACRO - a SAS REXX macro      */
/*          that invokes XEDIT              */
parse upper arg argstring
address command 'XEDIT' argstring
exit
```

The following program named SASSUB XEDIT shows how to temporarily pass control to SAS SUBSET:

```
/* SASSUB XEDIT - an XEDIT macro           */
/*          that enters SAS SUBSET         */
address sas '++SAS'
/* use CMSRETURN; to get back to XEDIT    */
exit rc
```

From the Program Editor window, submit the following statement:

```
sasxedit FILENAME FILETYPE FILEMODE;
```

This invokes SASXEDIT SASMACRO, which places you into an XEDIT session. At the XEDIT command line enter:

```
sassub
```

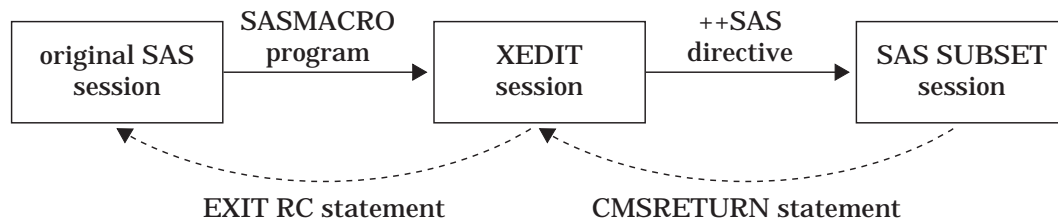
This invokes SASSUB XEDIT which puts you in SAS SUBSET in the Program Editor window as a continuation of your SAS session. Next submit this SAS statement:

```
cmsreturn;
```

This ends SAS SUBSET and resumes XEDIT, which runs the remaining portion of SASSUB XEDIT and then lets you continue your XEDIT session. Ending the XEDIT session (using FILE or QUIT) passes control back to the original SAS session, which runs the remaining portion of SASXEDIT SASMACRO and then returns you to the Program Editor window where you started.

Figure 9.1 on page 89 illustrates the flow of control among the original SAS session, the XEDIT session, and the SAS SUBSET session.

Figure 9.1 Flow of Control



Productivity Aids for Interactive SAS Sessions on CMS

A set of sample execs are provided with SAS to show how the SAS interface to REXX can be used to integrate CMS commands like FILELIST. Each of the following sample SASMACROs invokes the corresponding CMS command:

DIRLIST SASMACRO
 FILELIST SASMACRO
 MACLIST SASMACRO
 RDRLIST SASMACRO
 XEDIT SASMACRO

Like other SASMACROs, these can be issued as SAS statements without having to prefix them with X or CMS. In addition, you can issue commands from within CMS applications to interact directly with the SAS session. While any of the SASMACROs listed above are active, the following special commands are supported from within CMS applications:

DM *<command>*

is used on an XEDIT command line or an *xxxLIST* command line to switch the mode to the SAS windowing environment in a SAS subset and to execute a command. For example, DM KEYS switches to the KEYS window.

When you submit the CMSRETURN statement, control returns to the XEDIT session or *xxxLIST* menu. This command is implemented by the SAS\$DM exec.

INCLUDE

is used on a FILELIST file line to include the specified FILELIST file into the Program Editor window. Control returns to the FILELIST menu. This command is implemented by the SASSINCL exec.

PGM

is used on an XEDIT command line or an *xxx LIST* command line to switch the mode to the Program Editor window as a SAS subset. When you submit the CMSRETURN statement, control returns to the XEDIT session or *xxxLIST* menu. This command is implemented by the SAS\$PGM exec.

SUBMIT

is used on an XEDIT command line or a FILELIST file line to submit to SAS the current XEDIT file or specified FILELIST file. Control returns to the XEDIT session or FILELIST menu. This command is implemented by the SAS\$SUB exec.

Invoking a SASMACRO from the Windowing Environment Command Line

If you want to invoke SASMACROs directly from a command line:

- Specify the CMDMAC SAS system option.
- Include a command macro such as the following in an autoexec file:

```
%macro FILELIST/cmd parmbuff;
  pgm;
  submit "FILELIST &syspbuff;"
%mend;
```

You can then, for example, issue the following from the command line:

```
FILELIST * SAS A
```

Then, in the prefix area of the FILELIST screen, in front of a SAS program that you want to run, you can issue

```
SUBMIT /
```

This integration of the CMS commands with the SAS windowing environment enhances the usability of SAS under CMS.