# CHAPTER 15 Macros

## Macros in the OS/390 Environment

Most features of the SAS macro facility are portable. These features are documented in the *SAS Macro Language: Reference*. This chapter discusses the aspects of the macro facility that are specific to the OS/390 environment.

## Macro Variables

### Portable Automatic Macro Variables That Have Host-Specific Values

The following automatic macro variables are portable, but their values are host-specific:

SYSCC
contains the current SAS condition code that SAS will translate into a meaningful return code for OS/390 at the conclusion of the SAS session.

SYSDEVIC
contains the name of the current graphics device. The current graphics device is determined by the SAS system option DEVICE=. (See "DEVICE=" on page 337.) Ask your lcoal SAS Support Consultant which graphics devices are available at your site.

SYSENV
is provided for compatibility with the SAS System on other operating environments. Under OS/390, its value is FORE if you are running SAS under TSO; otherwise, its value is BACK. You cannot change the value of this variable.

SYSJOBID
contains the job name of the batch job that is currently executing, or the user ID that is associated with the current SAS session. SAS obtains this value from the TIOCNJOB field of the TIOT control block, except in the case of SAS/SESSION. With SAS/SESSION, SAS obtains the value from the User_id field that is returned by the Get_TP_Properties service of APPC/MVS. You cannot change the value of this variable.

SYSMAXLONG
returns the maximum long integer value allowed by OS/390, which is 2,147,483,647.

SYSRC
contains the return code from the most recent operating environment command that was issued from within a SAS session. The default value is 0.

SYSSCP
contains the operating environment abbreviation OS. You cannot change the value of this variable.

SYSSCPL
contains the operating environment name. For systems prior to OS/390 Release 1, SYSSCPL contains the value MVS. For OS/390 releases, SYSSCPL contains the value OS/390. You cannot change the value of this macro variable.

## OS/390-Specific Macro Variables

The following macro variables are available only under OS/390:

SYSDEXST
contains the value that is returned by the DSNEXST statement. (See "DSNEXST" on page 287.) SYSDEXST has a value of 1 if the data set specified in the DSNEXST statement is currently available, or a value of 0 if the data set is not currently available.

SYSJCTID
contains the value of the JCTUSER field of the JCT control block as mapped by the IEFAJCTB macro. It is a 7-byte character value.

SYSJMRID
contains the value of the JMRUSEID field of the JCT control block as mapped by the IEFAJMR macro. It is a 7-byte character value.

SYSUID
contains the value of the TSO userid that is associated with the SAS session, regardless of whether the session is a batch job, a remote connect session, a SAS/ SESSION connection, or a TSO session. SAS obtains this value from the ACEEUSRI field of the ACEE control block.

Four additional automatic macro variables that are available only under OS/390 can be used to help diagnose failures in dynamic allocation. Their values are updated each time SAS does a dynamic allocation as a result of a FILENAME or LIBNAME statement (or their equivalent DATA step or SCL functions). They are undefined until the first dynamic allocation is performed. These macro variables are:

SYS99ERR
: contains the error reason code that was returned in the SVC 99 request block.

SYS99INF
: contains the information reason code that was returned in the SVC 99 request block.

SYS99MSG
: contains the text of the message that is associated with the reason code.

SYS99R15
: contains the return code that was returned in R15 from SVC 99.

### Names to Avoid When Defining Automatic Macro Variables

When you define automatic macro variables, do not use names taken up by OS/390 reserved words, (see "Reserved OS/390 DDnames" on page 23) names of SAS system files (see Table 1.2 on page 16 ), or names beginning with &SYS. The prefix &SYS has been reserved for future use.

## Macro Statements

The following macro statements have OS/390-specific aspects:

%KEYDEF
: is analogous to the KEYDEF command in the windowing environment. It enables you to define function keys. The form of this statement is

    %KEYDEF <'>*key-name*<'> <'*definition*'>;

    The number of keys available depends on your terminal. Most terminals that are used under the OS/390 operating environment have either 12 or 24 function keys. To define a key, specify the *key-name* (F1 through F24) of the key and the new *definition*.

    If you omit the definition, SAS prints a message in the log showing the current definition of the key; otherwise, the key's definition is changed to whatever you specified.

%TSO
: executes TSO commands during an interactive SAS session. It is similar to the TSO statement. (See "TSO" on page 321.) The %TSO statement enables you to execute TSO commands immediately. It places the operating environment return code in the automatic variable SYSRC. You can use the %TSO statement either inside or outside a macro. The form of the statement is

    %TSO <*command*>;

    You can use any TSO command or any sequence of macro operations that generate a TSO command. If you omit the *command*, your SAS session is suspended and your OS/390 session is placed in TSO submode. To return to the SAS session, enter either RETURN or END.

If you execute a %TSO statement on an operating environment other than OS/390, the statement is treated as a comment.

%SYSEXEC
: executes TSO commands during an interactive SAS session. The form of the statement is

%SYSEXEC <*command*>;

Under OS/390, the %SYSEXEC statement works exactly like the %TSO statement. The two statements are different only if you transport your SAS program to a different operating environment. Because %SYSEXEC statements are recognized on multiple operating environments, each operating environment expects commands that are appropriate for that operating environment.

## Macro Functions

The following macro functions have OS/390-specific aspects:

%SCAN
: under OS/390 and other systems that use the EBCDIC collating sequence, if you specify no delimiters, SAS treats all of the following characters as delimiters:

blank . < ( + | & ! $ * ); ¬ – / , % ¦ ¢

%SYSGET
: has no effect under OS/390 except to generate the following message:

```
WARNING: The argument to macro function %SYSGET is
         not defined as a system variable.
```

## Autocall Libraries

An autocall library contains files that define SAS macros. Under OS/390, an autocall library is a partitioned data set. Each autocall macro should be a separate member in a partitioned data set. SAS Institute supplies some autocall macros in the system autocall library; you can also define autocall macros yourself in a user autocall library. In order to use the autocall facility, the SAS system option MAUTOSOURCE must be in effect. (See *SAS Language Reference: Dictionary* for details about MAUTOSOURCE.)

### Specifying a User Autocall Library

You can designate an physical file, or a concatenation of physical files, as your user-written autocall library in any of the following ways:

□ with the SASAUTOS= system option. You can designate one or more filerefs or data set names as your autocall library. See “SASAUTOS=” on page 391 for more information.

□ with the SASAUTOS parameter of the SAS CLIST (under TSO). In this case, SAS concatenates the user autocall library in front of the system autocall library, which is specified by the CLIST parameter MAUTS.

- with the SASAUTO= parameter of the SAS cataloged procedure.

## Example: Specifying an Autocall Library in Batch Mode

In batch mode, you could use the following JCL statements to specify an autocall library:

single autocall library:

```
//MYJOB    JOB account. ...
//         EXEC SAS,OPTIONS='MAUTOSOURCE'
//SASAUTOS DD DSN=MY.MACROS,DISP=SHR
```

concatenated autocall library:

```
//MYJOB    JOB account ...
//         EXEC SAS,OPTIONS='MAUTOSOURCE'
//SASAUTOS DD DSN=MY.MACROS1,DISP=SHR
//         DD DSN=MY.MACROS2,DISP=SHR
//         DD DSN=default.autocall.library,
//            DISP=SHR
```

## Example: Specifying an Autocall Library under TSO

Under TSO, you can specify an autocall library either when you invoke SAS or during a SAS session.

When you invoke SAS:

single autocall library:

```
sas options('mautosource sasautos=
   "myid.macros"')
```

concatenated autocall library:

```
sas options('mautosource sasautos=
   ("myid.macros1","myid.macros2",sasautos)')
```

During a SAS session:

single autocall library:

```
options mautosource sasautos=
   'myid.macros';
```

concatenated autocall library:

```
options mautosource sasautos=
   ('myid.macros1','myid.macros2',
   sasautos);
```

## Creating an Autocall Macro

To create an autocall macro, do the following:

1. Create a partitioned data set to function as an autocall library, or use an existing autocall library.
2. In the autocall library, create a member that contains the source statements for the macro. The member name must be the same as the name of the macro.

*Note:* The SAS macro facility allows you to include the underscore character in macro names; however, OS/390 does not allow the underscore character in partitioned data set member names. To create an autocall member for a macro name that contains an underscore, use a pound sign (#) in place of the underscore in the member name. For example, to create an autocall member for a macro named _SETUP_, name the member #SETUP#. However, invoke the macro by the macro name, as follows:

```
%_setup_
```

△

## Stored Compiled Macro Facility

The stored compiled macro facility gives you access to permanent SAS catalogs that contain compiled macros. In order for SAS to use stored compiled macros, the SAS system option MSTORED must be in effect. In addition, you use the SAS system option SASMSTORE= to specify the libref of a SAS data library that contains a catalog of stored compiled SAS macros. For more information about these options, see *SAS Language Reference: Dictionary*.

Using stored compiled macros offers the following advantages over other methods of making macros available to your session:

- SAS does not have to compile a macro definition when a macro call is made.
- Session-compiled macros and the autocall facility are also available in the same session.

Because you cannot re-create the source statements from a compiled macro, you must save the original macro source statements.

*Note:* Catalogs of stored compiled macros cannot be concatenated. △

If you don't want to use the stored compiled macro facility, you can make macros accessible to your SAS session or job by doing the following:

- placing all macro definitions in the program before calling them
- using a %INCLUDE statement to bring macro definitions into the program from external files*
- using the autocall facility to search predefined source libraries for macro definitions

Your most efficient choice may be to use the stored compiled macro facility.

### Accessing Stored Compiled Macros

The following example illustrates how to create a stored compiled macro in one session and then use the macro in a later session.

```
/*  Create a Stored Compiled Macro      */
/*      in One Session                  */
libname mylib 'u.macro.mysecret' disp=old;
options mstored sasmstore=mylib;
%macro myfiles / store;
  filename file1 'mylib.first';
  filename file2 'mylib.second';
```

* The %INCLUDE statement takes as arguments the same types of file specifications as the INCLUDE command. See "INCLUDE" on page 447.

```
%mend;

/*  Use the Stored Compiled Macro       */
/*       in a Later Session             */
libname mylib 'u.macro.mysecret' disp=shr;
options mstored sasmstore=mylib;

%myfiles
data _null_;
   infile file1;
      *statements that read input file FILE1;
   file file2;
      *statements that write to output file FILE2;
run;
```

## Other Host-Specific Aspects of the Macro Facility

### Collating Sequence for Evaluating Macro Characters

Under OS/390, the macro facility uses the EBCDIC collating sequence for %EVAL and for implicit evaluation of macro characters.

### SAS System Options Used by the Macro Facility

Table 15.1 on page 222 lists the SAS system options that are used by the macro facility and that have host-specific characteristics. It also tells you where to look for more information about these system options.

**Table 15.1** SAS System Options Used by the Macro Facility That Have Host-Specific Aspects

| **System Option** | **Description** | **See ...** |
|---|---|---|
| MSYMTABMAX= | specifies the maximum amount of memory available to all symbol tables (global and local combined). Under OS/390, the default value for this option is 1,048,576 bytes. | “System Options" chapter of the *SAS Companion for the OS/390 Environment* and the *SAS Language Reference: Dictionary* |
| MVARSIZE= | specifies the maximum number of bytes for any macro variable stored in memory (0 <= *n* <= 32768). Under OS/390, the default setting for this option is 8,192. | “System Options” chapter of the *SAS Companion for the OS/390 Environment*and the *SAS Language Reference: Dictionary* |
| SASAUTOS= | specifies the autocall library | “Specifying a User Autocall Library” and “Specifying a User Autocall Library” in this “Macros” chapter, and also the “System Options” chapter of the *SAS Companion for the OS/390 Environment* |

## Additional Sources of Information

For more information about the SAS macro facility, see the following documents:

- *SAS Macro Language: Reference*
- *SAS Language Reference: Dictionary*
- The online help for the macro facility. Enter **HELP MACRO** from a command line.

The correct bibliographic citation for this manual is as follows: SAS Institute Inc., *SAS® Companion for the OS/390 Environment, Version 8*, Cary, NC: SAS Institute Inc., 1999.

**SAS® Companion for the OS/390® Environment, Verison 8**

1–58025–523–X

SAS Institute Inc., SAS Campus Drive, Cary, North Carolina 27513.

1st printing, November 1999