CHAPTER 13 Functions and CALL Routines

Functions and CALL Routines in the OS/390 Environment

Portable functions are documented in *SAS Language Reference: Dictionary*. This chapter includes detailed information about the SAS functions and CALL routines that are specific to OS/390 or that have aspects specific to OS/390.

Dictionary

CALL SLEEP

Suspends execution of a SAS DATA step for a specified amount of time

OS/390 specifics: host call

Syntax

CALL SLEEP(*time*);

time
specifies the amount of time, in milliseconds (1/1,000 of a second), that you wish to suspend execution of a DATA step and the SAS process that is running that DATA step.

Details

CALL SLEEP puts the DATA step in which it is invoked into a nonactive wait state, using no CPU time and performing no input or output. If you are running multiple SAS processes, each process can execute CALL SLEEP independently without affecting the other processes.

In this example, the DATA step invokes CALL REPORT every hour:

```
data _null_;
   while (1);
      call report(a,b,c,d);
      call sleep(3600000);
   end;
run;
```

Note: In batch mode, extended sleep periods can trigger automatic host session termination based on timeout values set at your site. Contact your host system administrator as necessary to determine the timeout values used at your site. △

CALL SYSTEM

Issues an operating environment command during a SAS session

OS/390 specifics: all

Syntax

CALL SYSTEM(*command*);

command
can be a system command enclosed in quotes, an expression whose value is a system command, or the name of a character variable whose value is a system command. Under OS/390, "system command" includes TSO commands, CLISTs, and REXX execs.

Details

The CALL SYSTEM routine is similar to the X (or TSO) statement, the X (or TSO) command, the SYSTEM (or TSO) function, and the %SYSEXEC (or %TSO) macro statement.

In most cases, the X statement, the X command, or the %SYSEXEC macro statement are preferable because they require less overhead. However, the CALL SYSTEM routine can be useful in certain situations because it is executable and because it accepts expressions as arguments. For example, the following DATA step executes one of three CLISTs depending on the value of a variable named ACTION that is stored in an external file named USERID.TRANS.PROG:

```
data _null_;
  infile 'userid.trans.prog';

  /* action is assumed to have a value of    */
  /*    1, 2, or 3                           */
  /* create and initialize a 3-element array */
  input action;
  array programs{3} $ 11 c1-c3
    ("exec clist1" "exec clist2" "exec clist3");
  call system(programs{action});
run;
```

In this example, the array elements are initialized with character strings that consist of TSO commands for executing the three CLISTs. In the CALL SYSTEM statement, an expression is used to pass one of these character strings to the CALL SYSTEM routine. For example, if ACTION equals 2, then PROGRAMS{2}, which contains the EXEC CLIST2 command, is passed to the CALL SYSTEM routine.

Under OS/390, CALL TSO is an alias for the CALL SYSTEM routine.

See Also

- Statements: “TSO” on page 321 and “X” on page 323
- Functions: “SYSTEM” on page 202 and “TSO” on page 204
- Commands: “TSO” on page 449 and “X” on page 451
- “Macro Statements” on page 217

CALL TSO

Issues a TSO command or invokes a CLIST or a REXX exec during a SAS session

OS/390 specifics: all

Syntax

CALL TSO(*command*);

Details

The TSO and SYSTEM CALL routines are identical, with one exception: under an operating environment other than OS/390, the TSO CALL routine has no effect, whereas the SYSTEM CALL routine is always processed. See “CALL SYSTEM” on page 180 for more information.

DINFO

Returns information about a directory

OS/390 specifics: *info-item*

Syntax

DINFO (*directory-id, info-item*)

directory-id
specifies the identifier that was assigned when the directory was opened (generally by the DOPEN function).

info-item
specifies the name of the information item that is to be returned by the function.

Details

Directories opened with the DOPEN function are identified by a *directory-id* and have a number of associated information items. The DINFO, DOPTNAME, and DOPTNUM functions support the following directory information items under OS/390.

Table 13.1 Directory Information Items for UNIX System Services Directories

Item	Item Identifier	Definition
1	File Name	Directory name
2	Access Permission	Read, write, and execute permissions for owner, group, and other
3	Number of Links	Number of links in the directory
4	Owner Name	User ID of the owner

Item	Item Identifier	Definition
5	Group Name	Name of the owner's access group
6	Last Modified	Date contents last modified

Table 13.2 Directory Information Items for PDSs

Item	Item Identifier	Definition
1	Dsname	PDS name
2	Unit	Disk type
3	Volume	Volume on which data set resides
4	Disp	Disposition
5	Blksize	Block size
6	Lrecl	Record length
7	Recfm	Record format

Table 13.3 Directory Information Items for PDSEs

Item	Item Identifier	Definition
1	Dsname	PDSE name
2	Dsntype	Directory type
3	Unit	Disk type
4	Volume	Volume on which data set resides
5	Disp	Disposition
6	Blksize	Block size
7	Lrecl	Record length
8	Recfm	Record format

Example 1: UNIX System Services Directory Information

This first example generates output that includes information item names and values for a UNIX System Services directory:

```
data _null_;
   length opt $100 optval $100;

   /* Allocate directory */
   rc=FILENAME('mydir', '/u/userid');

   /* Open directory */
   dirid=DOPEN('mydir');

   /* Get number of information items */
   infocnt=DOPTNUM(dirid);
```

```
      /* Retrieve information items and */
      /* print to log                    */
      put @1 'Information for a UNIX
         System Services Directory:';
      do j=1 to infocnt;
         opt=DOPTNAME(dirid,j);
         optval=DINFO(dirid,upcase(opt));
         put @1 opt @20 optval;
      end;

      /* Close the directory */
      rc=DCLOSE(dirid);

      /* Deallocate the directory */
      rc=FILENAME('mydir');
   run;
```

Output 13.1 Example 1 Output: UNIX System Services Directory Information

```
Information for a UNIX System
   Services Directory:
File Name          /u/userid
Access Permission  drwxr-xr-x
Number of Links    17
Owner Name         MYUSER
Group Name         GRP
Last Modified      Apr 26 07:18

NOTE: The DATA statement used 0.09
   CPU seconds and 5203K.
```

Example 2: PDS Directory Information

This second example generates directory information for a PDSE:

```
data _null_;
   length opt $100 optval $100;

   /* Allocate directory */
   rc=FILENAME('mydir', 'userid.pdse.src');

   /* Open directory */
   dirid=DOPEN('mydir');

   /* Get number of information items */
   infocnt=DOPTNUM(dirid);

   /* Retrieve information items and */
   /* print to log                    */
   put @1 'Information for a PDSE:';
   do j=1 to infocnt;
      opt=DOPTNAME(dirid,j);
      optval=DINFO(dirid,upcase(opt));
      put @1 opt @20 optval;
   end;
   /* Close the directory */
   rc=DCLOSE(dirid);

   /* Deallocate the directory */
   rc=FILENAME('mydir');
run;
```

Output 13.2 Example 2 Output: PDSE Directory Information

```
Information for a PDSE:
Dsname             USERID.PDSE.SRC
Dsntype            PDSE
Unit               3380
Volume             ABC002
Disp               SHR
Blksize            260
Lrecl              254
Recfm              VB

NOTE: The DATA statement used 0.08
   CPU seconds and 5203K.
```

Example 3: PDS Directory Information

This example generates information item names and values for a PDS:

```
data _null_;
   length opt $100 optval $100;

   /* Allocate directory */
   rc=FILENAME('mydir', 'userid.mail.text');

   /* Open directory */
   dirid=DOPEN('mydir');

   /* Get number of information items */
   infocnt=DOPTNUM(dirid);

   /* Retrieve information items and */
   /* print to log                   */
   put @1 'Information for a PDS:';
   do j=1 to infocnt;
      opt=DOPTNAME(dirid,j);
      optval=DINFO(dirid,upcase(opt));
      put @1 opt @20 optval;
   end;

   /* Close the directory */
   rc=DCLOSE(dirid);

   /* Deallocate the directory */
   rc=FILENAME('mydir');
run;
```

Output 13.3 Example 3 Output: PDS Directory Information

```
Information for a PDS:
Dsname              USERID.MAIL.TEXT
Unit                3380
Volume              ABC005
Disp                SHR
Blksize             6160
Lrecl               80
Recfm               FB

NOTE: The DATA statement used 0.07
   CPU seconds and 5211K.
```

See Also

- “DOPEN” on page 186
- “DOPTNAME” on page 186
- “DOPTNUM” on page 187
- *SAS Language Reference: Dictionary*

DOPEN

Opens a directory and returns a directory identifier value

OS/390 specifics: file systems

Syntax

DOPEN (*fileref*)

fileref
specifies the directory to be opened.

Details

DOPEN applies to directory structures, which are available in partitioned data sets (PDS, PDSE) and in UNIX System Services. For code examples, see “DINFO” on page 182.

See Also

- “DOPTNAME” on page 186
- “DOPTNUM” on page 187
- *SAS Language Reference: Dictionary*

DOPTNAME

Returns the name of a directory information item

OS/390 specifics: *info-item*

Syntax

DOPTNAME (*directory-id*,*info-item*)

directory-id
specifies the identifier that was assigned when the directory was opened (generally by the DOPEN function).

info-item
specifies the number of a directory information item. For definitions of information item numbers and code examples, see"DINFO" on page 182.

Details

The DOPTNAME function returns the name of the specified information item number for a file that was previously opened with the DOPEN function.

See "DINFO" on page 182 for information item numbers and definitions and code examples.

See Also

- "DOPEN" on page 186
- "DOPTNUM" on page 187
- *SAS Language Reference: Dictionary*

DOPTNUM

Returns the number of information items available for a directory

OS/390 specifics: return value

Syntax

DOPTNUM (*directory-id*)

directory-id
specifies the identifier that was assigned when the directory was opened.

Details

Currently, the number of information items available for a PDS directory is 7, for a PDSE directory is 8, and for a UNIX System Services directory is 7.

For code examples, see "DINFO" on page 182.

See Also

- □ "DOPEN" on page 186
- □ "DOPTNAME" on page 186
- □ *SAS Language Reference: Dictionary*

FCLOSE

Closes an external file, a directory, or a directory member

OS/390 specifics: file close is strongly recommended

Syntax

FCLOSE (*file-id*)

file-id
is the file-identifier that was assigned when the file was opened.

Details

Files opened with the FOPEN function are not closed automatically after processing. All files opened with FOPEN should be closed with FCLOSE. For code examples, see "FINFO" on page 191

See Also

- □ "FOPEN" on page 196

FDELETE

Deletes an external file

OS/390 specifics: *fileref*

Syntax

FDELETE (*fileref*)

fileref
identifies an external file. The fileref must have been previously associated with a sequential file, a PDS, a PDSE, or a UNIX System Services file using a FILENAME statement or FILENAME function. The fileref cannot represent a concatenation of multiple files.

Details

If the fileref specified with FDELETE is associated with a UNIX System Services directory, PDS, or PDSE, then that directory, PDS, or PDSE must be empty. The user that calls FDELETE must also have appropriate privilege to delete the directory or file.

Example

```
filename delfile 'myfile.test';
  data _null_;
  rc=fdelete('delfile');
  run;
```

See Also

- *SAS Language Reference: Dictionary*

FEXIST

Verifies the existence of an external file associated with a fileref and returns a value

OS/390 specifics: *fileref*

Syntax

FEXIST(*fileref*)

fileref
identifies an external file. Under OS/390, it can be a fileref or any valid DDname that has been previously associated with an external file using either a TSO ALLOCATE command or a JCL DD statement.

See Also

- *SAS Language Reference: Dictionary*

FILEEXIST

Verifies the existence of an external file by its physical name and returns a value

OS/390 specifics: *file-name*

Syntax

FILEEXIST(*filename*)

filename
is a fully qualified operating environment data set name or a fully qualified path (for UNIX System Services files).

See Also

- *SAS Language Reference: Dictionary*

FILENAME

Assigns or deassigns a fileref for an external file, a directory, or an output device and returns a value

OS/390 specifics: host options, devices

Syntax

FILENAME (*fileref,filename<,device <,host-options>>*)

fileref
specifies the fileref to assign to an external file.

filename
specifies the external file. Specifying a blank *file-name* deassigns one that was previously assigned.

device
specifies the type of device if the fileref points to an output device rather than to a physical file:

DUMMY
output to the file is discarded

PIPE
an unnamed pipe

PLOTTER
an unbuffered graphics output device

PRINTER
a printer or printer spool file

TERMINAL
the user's terminal

TAPE
a tape driver

host-options

are host-specific options that may be specified in the FILENAME statement. These options can be categorized into several groups. For details, see the following sections:

- “FILENAME” on page 294
- “DCB Attribute Options” on page 300
- “SYSOUT Data Set Options for the FILENAME Statement” on page 304
- “Subsystem Options for the FILENAME Statement” on page 306
- “Options That Specify SMS Keywords” on page 303
- “Host-Specific Options for UNIX System Services Files” on page 96 .

You can specify host options in any order following the file specification and the optional *device* specification. When specifying more than one option, use a blank space to separate each option. Values for options may be specified with or without quotes. However, if a value contains one of the supported national characters ($, #, or @), the quotes are required.

See Also

- FILENAME statement, see “FILENAME” on page 294
- *SAS Language Reference: Dictionary*

FILEREF

Verifies that a fileref has been assigned for the current SAS session and returns a value

OS/390 specifics: *fileref*

Syntax

FILEREF (*fileref*)

fileref

specifies the fileref to be validated. Under OS/390, *fileref* can be a DDname that was assigned using the TSO ALLOCATE command or JCL DD statement.

See Also

- *SAS Language Reference: Dictionary*

FINFO

Returns the value of a file information item

OS/390 specifics: *info-item*

Syntax

FINFO (*file-id*,*info-item*)

file-id
specifies the identifier that was assigned when the file was opened (generally by the FOPEN function).

info-item
specifies the number of the information item that is to be retrieved.

Details

The FINFO function returns the value of a specified information item for an external file that was previously opened and assigned a *file-id* by the FOPEN function.

The FINFO, FOPTNAME, and FOPTNUM functions support the following information items.

Table 13.4 Information Items for Unix System Services Files

Item	Item Identifier	Definition
1	File Name	File name
2	Access Permission	Read, write, and execute permissions for owner, group, and other
3	Number of Links	Number of links in the file
4	Owner Name	User ID of the owner
5	Group Name	Name of the owner's access group
6	File Size	File size
7	Last Modified	Date file last modified

Table 13.5 Information Items for Sequential Files and members of PDSs and PDSEs

Item	Item Identifier	Definition
1	Dsname	File name
2	Unit	Disk type
3	Volume	Volume on which data setresides
4	Disp	Disposition
5	Blksize	Block size
6	Lrecl	Record length
7	Recfm	Record format

Example 1: Sequential File Information

The following example generates output that shows the information items available for a sequential data set:

```
data _null_;
   length opt $100 optval $100;

   /* Allocate file  */
   rc=FILENAME('myfile',
      'userid.test.example');
      /* Open file */
      fid=FOPEN('myfile');

      /* Get number of information
         items */
      infocnt=FOPTNUM(fid);

      /* Retrieve information items
         and print to log  */
      put @1 'Information for a
         Sequential File:';
      do j=1 to infocnt;
         opt=FOPTNAME(fid,j);
         optval=FINFO(fid,upcase(opt));
         put @1 opt @20 optval;
      end;

      /* Close the file */
      rc=FCLOSE(fid);

      /* Deallocate the file */
      rc=FILENAME('myfile');
   run;
```

Output 13.4 Example 1 Output: Sequential File Information

```
Information for a Sequential File:
Dsname             USERID.TEST.EXAMPLE
Unit               3380
Volume             ABC010
Disp               SHR
Blksize            23392
Lrecl              136
Recfm              FB

NOTE: The DATA statement used 0.10
   CPU seconds and 5194K.
```

Example 2: PDS, PDSE Member Information

This next example shows the information items availabile for PDS and PDSE members:

```
data _null_;
  length opt $100 optval $100;

  /* Allocate file  */
  rc=FILENAME('myfile',
     'userid.test.data(oats)');

  /* Open file */
  fid=FOPEN('myfile');

  /* Get number of information
     items */
  infocnt=FOPTNUM(fid);
```

```
    /* Retrieve information items
          and print to log */
    put @1 'Information for a PDS
       Member:';
    do j=1 to infocnt;
      opt=FOPTNAME(fid,j);
      optval=FINFO(fid,upcase(opt));
      put @1 opt @20 optval;
    end;

    /* Close the file */
    rc=FCLOSE(fid);

    /* Deallocate the file */
    rc=FILENAME('myfile');
  run;
```

Output 13.5 Example 2 Output: PDS, PDSE Member Information

```
Information for a PDS Member:
Dsname             USERID.TEST.DATA(OATS)
Unit               3380
Volume             ABC006
Disp               SHR
Blksize            1000
Lrecl              100
Recfm              FB

NOTE: The DATA statement used 0.05
   CPU seconds and 5194K.
```

Example 3: UNIX System Services File Information

This final example shows the information items available for UNIX System Services files:

```
  data _null_;
     length opt $100 optval $100;

     /* Allocate file  */
     rc=FILENAME('myfile',
        '/u/userid/one');

     /* Open file */
     fid=FOPEN('myfile');

     /* Get number of information
        items */
     infocnt=FOPTNUM(fid);

     /* Retrieve information items
        and print to log */
     put @1 'Information for a UNIX
        System Services File:';
     do j=1 to infocnt;
```

```
      opt=FOPTNAME(fid,j);
      optval=FINFO(fid,upcase(opt));
      put @1 opt @20 optval;
   end;

   /* Close the file */
   rc=FCLOSE(fid);

   /* Deallocate the file */
   rc=FILENAME('myfile');
run;
```

Output 13.6 Example 3 Output: UNIX System Services File Information

```
Information for a UNIX
   System Services File:
File Name          /u/userid/one
Access Permission  ---rw-rw-rw-
Number of Links    1
Owner Name         USERID
Group Name         GRP
File Size          4
Last Modified      Apr 13 13:57

NOTE: The DATA statement used
   0.07 CPU seconds and 5227K.
```

See Also

□ "FCLOSE" on page 188
□ "FOPEN" on page 196
□ "FOPTNAME" on page 197
□ "FOPTNUM" on page 198
□ *SAS Language Reference: Dictionary*

FOPEN

Opens an external file and returns a file identifier value

OS/390 specifics: files opened with FOPEN must be explicitly closed with FCLOSE

Syntax

FOPEN (*fileref* <,*open-mode* <,*record-length* <,*record-format*>>>)

fileref
specifies the fileref assigned to the external file.

open-mode
specifies the type of access to the file:

A	APPEND mode allows writing new records after the current end of the file.

I	INPUT mode allows reading only (default).
O	OUTPUT mode defaults to the OPEN mode specified in the host option in the FILENAME statement or function. If no host option is specified, it allows writing new records at the beginning of the file.
S	Sequential input mode is used for pipes and other sequential devices such as hardware ports.
U	UPDATE mode allows both reading and writing.

record-length
specifies the logical record length of the file. To use the existing record length for the file, specify a length of 0, or do not provide a value here.

record-format
specifies the record format of the file. To use the existing record format, do not specify a value here. Valid values are as follows:

B	data are to be interpreted as binary data.
D	use default record format.
E	use editable record format.
F	file contains fixed length records.
P	file contains printer carriage control in host-dependent record format. For data sets with FBA or VBA record format, specify ‘P’ for the *record-format* argument.
V	file contains variable length records.

Details

Under OS/390, files that have been opened with FOPEN must be closed with FCLOSE at the end of a DATA step; files are not closed automatically after processing.

See “FINFO” on page 191 for code examples.

See Also

- “FCLOSE” on page 188
- “FOPTNAME” on page 197
- “FOPTNUM” on page 198
- *SAS Language Reference: Dictionary*

FOPTNAME

Returns the name of an item of information about a file

OS/390 specifics: *info-item*

Syntax

FOPTNAME (*file-id*,*info-item*)

file-id
specifies the identifier that was assigned when the file was opened (generally by the FOPEN function).

info-item
specifies the information item whose name is to be returned by the function.

Details

FOPTNAME returns the name of a specified information item associated with the specified *file-id*. The *file-id* is assigned when the file is opened with the FOPEN function.

For definitions of information item numbers and code examples, see “FINFO” on page 191.

See Also

- □ “FCLOSE” on page 188
- □ “FOPEN” on page 196
- □ “FOPTNUM” on page 198
- □ *SAS Language Reference: Dictionary*

FOPTNUM

Returns the number of information items that are available for a file

OS/390 specifics: return value

Syntax

FOPTNUM (*file-id*)

file-id
specifies the identifier that was assigned when the file was opened (generally by the FOPEN function).

Details

Currently, the number of information items available for a sequential file, a PDS member, and a UNIX System Services file is 7.

For code examples, refer to “FINFO” on page 191.

See Also

- “FCLOSE” on page 188
- “FOPEN” on page 196
- “FOPTNAME” on page 197
- *SAS Language Reference: Dictionary*

HOSTHELP

Invokes the native help system to display or close the specified help information

OS/390 specifics: not supported

KTRANSLATE

Replaces specific characters in a character expression

OS/390 specifics: to/from pairs

Syntax

KTRANSLATE(*source, to-1, from-1<to-2, from-2...to-n, from-n>*)

Details

In the OS/390 environment, KTRANSLATE requires a *from* argument for each *to* argument. Also, there is no practical limit to the number of to/from pairs you can specifiy.

KTRANSLATE differs from TRANSLATE in that it supports single-byte character set replacement by double-byte characters, or vice versa.

See Also

- “TRANSLATE” on page 203
- *SAS Language Reference: Dictionary*

LIBNAME

Assigns or deassigns a libref for a SAS data library and returns a value

OS/390 specifics: *libref, SAS-data-library*

Syntax

LIBNAME (*libref*, <,*SAS-data-library* <,*engine* <,*options*>>>)

libref
specifies the libref to assign to a SAS data library.

SAS-data-library
specifies the SAS data library.

Details

If no value is provided for *SAS-data-library* or if *SAS-data-library* has a value of `''`(with no blank space), LIBNAME dissociates the libref from the data library. If the operation is successful, the return value is zero.

Under TSO, DDnames (assigned by the TSO ALLOCATE command) can also be used to refer to SAS data libraries.

Example

```
rc=libname('v7dat','myapp.demo.v7dat', 'v7');
```

See Also

- LIBNAME statement, see “LIBNAME” on page 313
- *SAS Language Reference: Dictionary*

MOPEN

Opens a file by directory ID and member name and returns either the file identifier or a zero

OS/390 specifics: file systems

Syntax

MOPEN (*directory-id ,member-name<open-mode* <,*record-length* <,*record-format*>>>)

Details

MOPEN applies to files in directory structures, which are available in partitioned data sets (PDS, PDSE) and in UNIX System Services.

Under OS/390, MOPEN can open files for output and append.

See Also

- “DOPEN” on page 186
- *SAS Language Reference: Dictionary*

PATHNAME

Returns the physical name of a SAS data library or of an external file or returns a blank

OS/390 specifics: *fileref, libref*

Syntax

PATHNAME (*fileref* | *libref*)

fileref
specifies the fileref that was assigned to an external file.

libref
specifies the libref assigned to a SAS data library.

Details

When PATHNAME is applied to a concatenation, it returns a list of data set names encoded in parentheses.

Under OS/390, you can also use any valid DDname previously allocated using a TSO ALLOCATE command or a JCL DD statement.

See Also

- *SAS Language Reference: Dictionary*

SYSGET

Returns the value of a specified host-environment variable

OS/390 specifics: *host-variable*

Syntax

SYSGET (*host-variable*)

host-variable
is the name of one of the parameters defined in the CLIST by which SAS was invoked

Details

If the variable specified was not included in the SAS invocation, you receive a “NOTE: Invalid argument to the function SYSGET” and _ERROR_ is set to 1.

Example

The following example returns the system options specified in the OPTIONS parameter of the SAS CLIST and prints to the specified log.

```
data _null_;
  opstr=sysget('OPTIONS');
  if _ERROR_ then put 'no options supplied';
  else put 'options supplied are:' optstr;
run;
```

See Also

- *SAS Language Reference: Dictionary*

SYSTEM

Issues an operating environment command during a SAS session

OS/390 specifics: *command*, related commands, statements, macros

Syntax

SYSTEM(*command*)

command
can be a system command enclosed in quotes, an expression whose value is a system command, or the name of a character variable whose value is a system command. Under OS/390, the term system command refers to TSO commands, CLISTs, and REXX execs.

Details

The SYSTEM function is similar to the X (or TSO) statement, the X (or TSO) command, the CALL SYSTEM (or CALL TSO) routine, and the %SYSEXEC (or %TSO) macro statement. In most cases, the X statement, the X command, or the %SYSEXEC macro statement are preferable because they require less overhead.

This function returns the operating environment return code after the command, CLIST, or REXX exec is executed.

SAS executes the SYSTEM function immediately. Under OS/390, TSO is an alias for the SYSTEM function. On other operating environments, the TSO function has no effect, whereas the SYSTEM function is always processed.

You can use the SYSTEM function to issue most TSO commands or to execute CLISTs or REXX execs. However, you cannot issue the TSO commands LOGON and LOGOFF, and you cannot execute CLISTs that include the TSO ATTN statement.

Example 1

In the following example, the SYSTEM function is used to allocate an external file:

```
data _null_;
   rc=system('alloc f(study) da(my.library)');
run;
```

For a fully qualified data set name, use the following statements:

```
data _null_;
   rc=system("alloc f(study)
      da('userid.my.library')");
run;
```

Example 2

In the second example, notice that the command is enclosed in double quotes. When the TSO command includes quotes, it is best to enclose the command in double quotes instead of single quotes. If you choose to use single quotes, then double each quote in the TSO command:

```
data _null_;
   rc=system('alloc f(study)
       da(''userid.my.library'')');
run;
```

See Also

- □ Statements: "TSO" on page 321 and "X" on page 323
- □ CALL routines: "CALL SYSTEM" on page 180 and "CALL TSO" on page 181
- □ Commands: "TSO" on page 449 and "X" on page 451
- □ "Macro Statements" on page 217

TRANSLATE

Replaces specific characters in a character expression

OS/390 specifics: to/from pairs required

Syntax

TRANSLATE (*source, to-1, from-1,* < . . . *to-n, from-n*>)

Details

In the OS/390 environment, TRANSLATE requires a *from* argument for each *to* argument. Also, there is no practical limit to the number of to/from pairs you can specifiy.

TRANSLATE handles character replacement for single-byte character sets only. See KTRANLSATE to replace single-byte characters with double-byte characters, or vice versa.

See Also

- □ "KTRANSLATE" on page 199
- □ *SAS Language Reference: Dictionary*

TSO

Issues a TSO command or invokes a CLIST or a REXX exec during a SAS session

OS/390 specifics: all

Syntax

TSO(*command*)

Description

The SYSTEM and TSO functions are identical, with one exception: under an operating environment other than OS/390, the TSO function has no effect, whereas the SYSTEM function is always processed. See "SYSTEM" on page 202 for more information.

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

Copyright © 1999 by SAS Institute Inc., Cary, NC, USA.

1–58025–523–X

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

1st printing, November 1999