CHAPTER
14

Functions and CALL Routines

Functions and CALL Routines in the CMS Environment **137**
BYTE **137**
CMS **140**
DINFO **142**
DOPEN **144**
DOPTNAME **145**
DOPTNUM **145**
FCLOSE **146**
FDELETE **147**
FEXIST **147**
FILEEXIST **148**
FILENAME **148**
FILEREF **150**
FINFO **150**
FOPEN **153**
FOPTNAME **154**
FOPTNUM **155**
GETEXEC **155**
KTRANSLATE **156**
LIBNAME **157**
MOPEN **158**
PATHNAME **158**
RANK **159**
SYSGET **159**
SYSTEM **160**
TRANSLATE **160**

Functions and CALL Routines in the CMS Environment

Portable SAS functions and CALL routines are documented in *SAS Language Reference: Dictionary*. This section includes detailed information about only those SAS functions and CALL routines that are CMS-specific or that have syntax or behavior that is specific to the CMS operating environment.

BYTE

Returns one character in the EBCDIC collating sequence

CMS specifics: EBCDIC collating sequence

Syntax

BYTE (*n*)

n
is an integer that represents a particular EBCDIC character. Its value must be between 0 and 255.

Details

Under CMS, the BYTE function returns the *n*th character in the EBCDIC collating sequence. Some operating environments return the corresponding character in the ASCII collating sequence for this function.

Example

The following DATA step uses the BYTE function:

```
data;
   x=byte(193);
run;
```

The example results in *x* being set to the character A (hexadecimal 'C1'x).

See Also

- *SAS Language Reference: Dictionary*

CALL PUTEXEC

Assigns a value to an EXEC variable

CMS specifics: all

Syntax

CALL PUTEXEC ('*argument1*', *argument2*);

'argument1'
is the name of an EXEC variable. This argument must be uppercased and enclosed in single quotes.

argument2
is the value to be assigned. This argument can be specified as a literal or a variable name. If you specify a literal, it must be enclosed in single quotes.

CMS Specifics

The PUTEXEC CALL routine assigns a value to an EXEC variable. The EXEC must be written in EXEC2 or REXX. To use the PUTEXEC CALL routine effectively, you must be familiar with one of these languages.

If the variable that you specify as *argument1* does not currently exist as an EXEC variable, it is created.

Here are some points to consider when using the PUTEXEC CALL routine:

- All EXEC variable names specified as arguments for PUTEXEC must be uppercased. This is true for REXX as well as EXEC2, even though REXX makes no distinction between upper- and lowercase in its variable names. If you specify a REXX compound symbol, such as NAME.1, as an argument, only the stem (the part of the name preceding the period) must be in uppercase.
- Although EXEC2 variables always begin with an ampersand (&), you must omit the ampersand when you use the name as an argument.
- The CMS EXEC processor hides all variables except those in the current EXEC. Hidden variables are not accessible by PUTEXEC.
- Arguments to PUTEXEC must be character values, and the values of EXEC variables, even numeric values, are always retained in character format. By default, SAS converts values automatically from numeric to character or character to numeric. However, it is better to use the DATA step functions INPUT and PUT to convert character values from EXEC variables to a specific numeric format, and vice versa.
- If the interface to EXEC variables fails, PUTEXEC assumes that its arguments are invalid (because no EXEC variables can be accessed), and a message is written to the SAS log.

See “Using the PUTEXEC CALL Routine” on page 98 for information and examples. Also, refer to the IBM *VM/ESA REXX/VM Reference* and *VM/ESA REXX/VM User’s Guide*. If you are not familiar with EXEC2, refer to the IBM *VM/SP EXEC 2 Reference*.

Example

The following statement specifies the literal USR191 as the value to assign to the EXEC variable LABEL:

```
call putexec('LABEL','USR191');
```

CALL SLEEP

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

CMS specifics: host call

Syntax

CALL SLEEP(*time*);

time

specifies the number of milliseconds (1/1,000 of a second) you want 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 non-active 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: 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. △

CMS

Invokes a CMS or CP command and returns the return code

Alias: SYSTEM

CMS specifics: all

Syntax

CMS (’*command*’)

’command’
is a character string that corresponds to a CMS or CP command.

Details

The CMS function invokes one CMS or CP command and returns the return code that was set by execution of the command. Because the CMS or SYSTEM function is part of an executable SAS statement, you can conditionally execute certain CMS and CP commands within a SAS session (unlike the CMS statement) and to use the CMS SUBCOM facility for support.

For example, you can use the CMS function in the following assignment statement:

```
rc=CMS('command');
```

rc is a variable that contains the return code that was set by execution of the command.

Commands invoked with the CMS function are executed when the DATA step executes, not when the statement containing the function is scanned.

The limitations on CMS and CP commands in CMS subset mode also apply to the CMS function; that is, commands that use the user area are not allowed.

Example

The following SAS program determines whether today's data file exists on an accessed minidisk. It does this using the SAS functions WEEKDAY and TODAY and by issuing a CMS STATE command. The program references five CMS files; each file contains data for one weekday.

```
DAY1 DATA A
DAY2 DATA A
DAY3 DATA A
DAY4 DATA A
DAY5 DATA A
```

The first DATA step uses information from functions TODAY and WEEKDAY in the CMS STATE command, which determines whether a data file such as DAY2 DATA * exists. If the file exists (rc=0), the program creates a macro variable called &DAILY. &DAILY contains the name of the file that exists and is used in the INFILE statement in the second DATA step. If the file does not exist, the SAS program aborts and the second DATA step is never executed.

```
data day;
   wd=weekday(today())-2;
   if wd=0 then wd=5;
   number=put(wd,1.);
   rc=cms('state day'||number||' data *');
      if rc=0 then do;
         name=' "day'||number||' data" ';
         call symput('daily',name);
         end;
      else do;
         put ' The data file does not exist.';
         abort return;
         end;
run;
data daily;
   infile &daily;
   input branch $ 1-20 dept 22-24 @26 revenue 10.;
run;
```

The sample program illustrates the difference between a command invoked with the CMS function and a command invoked with the CMS statement or in CMS subset mode. Commands that are invoked with the CMS function are not invoked until execution time, after all statements in a step have been scanned. Commands that are invoked with a CMS statement or in CMS subset mode are executed when SAS encounters them while scanning the step.

See Also

- "Issuing CMS and CP Commands during a SAS Session" on page 9

COLLATE

Generates an EBCDIC collating sequence character string

CMS specifics: EBCDIC collating sequence

Syntax

COLLATE (*start-character* <,*end-character*> <,*length*>)

start-character
is an integer that corresponds to the beginning character in a sequence.

end-character
is an integer that corresponds to the last character in a sequence. If you omit *end-character*, you must mark its place with a comma.

length
is the number of characters specified in a string.

Details

Under CMS, the COLLATE function returns a string of characters from the EBCDIC collating sequence. Some operating environments return the corresponding character in the ASCII collating sequence for this function.

If neither the end character nor the length is specified, COLLATE returns a character string up to 200 characters long. If both the end character and the length are specified, the length is ignored.

See Also

- *SAS Language Reference: Dictionary*

DINFO

Returns information from the directory of an aggregate external file

CMS specifics: *info-item*

Syntax

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

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

info-item
specifies the information item to be retrieved. Under CMS, the single valid value for *info-item* is DIRECTORY, which returns the directory path associated with *directory-id*. If *directory-id* represents a single aggregate, the information returned by DINFO will be of the form '*pathname*'. If *directory-id* represents a concatenated series of aggregates, the information returned by DINFO will be of the form ('*pathname_1*', ... '*pathname_n*').

DINFO Output for SFS Directories

The following example and output illustrate the use of DINFO and the other directory access functions for SFS directories:

```
data _null_;
   length opt $100 optval $100;
   rc=FILENAME('mydir', 'user1.');  /* allocate directory     */
   dirid = DOPEN('mydir');          /* open directory         */
   infocnt=DOPTNUM(dirid); /* get number of information items */

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

   rc = DCLOSE(dirid);            /* close the directory      */
   rc = FILENAME('mydir');        /* deallocate the directory */
run;
```

Output 14.1 DINFO Output for an SFS Directory

```
Information for an SFS Directory:
Directory          SFSFP:USER1 .
NOTE: DATA statement used:
      real time           0.85 seconds
      cpu time            0.28 seconds
```

DINFO Output for Minidisks

The following example and output illustrate the use of DINFO and the other directory access functions for minidisks:

```
data _null_;
   length opt $100  optval $100;
   rc=FILENAME('mydir', 'A'); /* allocate directory */
   dirid = DOPEN('mydir'); /* open directory */
   infocnt=DOPTNUM(dirid); /* Get number of information items */

   /* Retrieve information items and print to log*/
   put @1 'Information for a Minidisk Directory:';
   do j=1 to infocnt;
```

```
      opt = DOPTNAME(dirid,j);
      optval  = DINFO(dirid,upcase(opt));
      put @1 opt @20 optval;
   end;

   rc = DCLOSE(dirid);  /* close the directory */
   rc = FILENAME('mydir'); /* deallocate the directory */
run;
```

Output 14.2 DINFO Output for a Minidisk

```
Information for a Minidisk Directory:
Directory          A
NOTE: DATA statement used:
      real time           0.18 seconds
      cpu time            0.18 seconds
```

See Also

- "DOPEN" on page 144
- "DOPTNAME" on page 145
- "DOPTNUM" on page 145
- *SAS Language Reference: Dictionary*

DOPEN

Opens an aggregate external file and returns a directory identifier value

CMS specifics: directory specification

Syntax

DOPEN ('*fileref*')

fileref
specifies the aggregate external file to be opened. Single quotation marks around the fileref are required unless the literal fileref name is represented by a macro variable or a data step variable.

Details

You must associate a fileref with the aggregate before calling DOPEN. See “DINFO” on page 142 for an example that shows the use of DOPEN and other directory access functions.

CMS MACLIBs are currently not supported by the DOPEN function.

See Also

- “DOPTNAME” on page 145
- “DOPTNUM” on page 145
- *SAS Language Reference: Dictionary*

DOPTNAME

Returns the name of a directory information item

CMS specifics: *info-item*

Syntax

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

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

info-item
specifies the number of the information item, the name of which will be returned by the function. For DOPTNAME, the single valid value for *info-item* is 1, which will return the information item name DIRECTORY for a valid *directory-id*.

Details

See “DINFO” on page 142 for an example that shows the use of DOPTNAME and other functions.

See Also

- “DOPEN” on page 144
- “DOPTNUM” on page 145
- *SAS Language Reference: Dictionary*

DOPTNUM

Returns the number of information items available for a directory

CMS specifics: number of information items

Syntax

DOPTNUM (*directory-id*)

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

Details

DOPTNUM returns a number that represents the total number of information items available for a given *directory-id*. Currently, the information item named DIRECTORY is the sole information item available for directories, hence DOPTNUM currently returns a value of 1 for a valid *directory-id*.

See “DINFO” on page 142 for an example that shows the use of DOPTNUM and other functions.

See Also

- □ “DOPEN” on page 144
- □ “DOPTNAME” on page 145
- □ *SAS Language Reference: Dictionary*

FCLOSE

Closes an external file, a directory, or a directory member and returns a value

CMS specifics: FCLOSE is required

Syntax

FCLOSE (*file-id*)

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

Details

Under CMS, you must close files with the FCLOSE function at the end of a DATA step; files are not closed automatically after processing.

See “FINFO” on page 150 for an example that demonstrates the use of the FOPEN, FCLOSE, FINFO, FILENAME, FOPTNAME and FOPTNUM functions.

See Also

- "FINFO" on page 150
- "FOPEN" on page 153
- "FOPTNAME" on page 154
- "FOPTNUM" on page 155
- *SAS Language Reference: Dictionary*

FDELETE

Deletes an external file or an empty directory

CMS specifics: *fileref*

Syntax

FDELETE('*fileref*')

fileref
specifies the fileref that you assign to the external file or SFS directory. Single quotation marks around the fileref are required unless the literal fileref name is represented by a macro variable or a data step variable. You can assign filerefs by using the FILENAME statement or the FILENAME function.

Details

To delete a directory, you must have authorization to do so.

In the CMS environment, the fileref cannot be assigned by using a CMS FILEDEF command.

See Also

- "FILENAME" on page 148
- *SAS Language Reference: Dictionary*

FEXIST

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

CMS specifics: *fileref*

Syntax

FEXIST(*fileref*)

fileref
specifies the fileref assigned to an external file.

Details

The fileref must have been previously assigned.

See Also

- □ "FILEEXIST" on page 148
- □ "FILENAME" on page 148
- □ "FILEREF" on page 150
- □ *SAS Language Reference: Dictionary*

FILEEXIST

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

CMS specifics: *filename*

Syntax

FILEEXIST('*filename*')

filename
is a fully qualified physical filename of the external file. Single quotation marks around the filename are required unless the literal filename is represented by a macro variable or a data step variable.

Details

In the CMS environment, the filename cannot be a physical name that was assigned by using an environment variable. See "SYSGET" on page 159 for information on retrieving environment variables.

See Also

- □ "FEXIST" on page 147
- □ "FILENAME" on page 148
- □ "FILEREF" on page 150
- □ *SAS Language Reference: Dictionary*

FILENAME

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

CMS specifics: host options, devices, *dir-ref*

Syntax

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

fileref
specifies the fileref to assign to an external file.

filename
specifies the external file. Specifying a blank *filename* 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:

DISK
specifies that the file is to be read from or written to disk. This is the default, which does not have to be specified. This device type uses the native CMS interface.

DUMMY
specifies that output to the external file is to be discarded.

PIPE
a CMS pipeline

PRINTER
a printer or printer spool file

PUNCH
associates the fileref with the virtual PUNCH. This device is for output only. Specifying this device type causes the CMS SAS interface to issue a FILEDEF for the device. Use the CP SPOOL and TAG commands to control the destination.

READER
specifies that the file is to be a reader file. The BLKSIZE= option is ignored for this device because CMS does not support blocking. This device is for input only. Specifying this device type causes the CMS SAS interface to issue a FILEDEF for the device. If you have more than one file in your reader, the one that is ordered first is read. Use the CP ORDER READER and CP QUERY READER commands to determine the order of your reader files.

TERMINAL
the user's terminal

TAPE
a tape drive.

host-options
are host-specific options that may be specified in the FILENAME statement. See "FILENAME" on page 227.

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.

dir-ref
specifies the fileref that is assigned to the directory in which the external file resides.

See Also

- □ *SAS Language Reference: Dictionary*

FILEREF

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

CMS specifics: *fileref*

Syntax

FILEREF (*fileref*)

fileref
specifies the fileref to be validated. Under CMS, *fileref* can be a DDname that was assigned using the CMS FILEDEF command.

See Also

- □ *SAS Language Reference: Dictionary*

FINFO

Returns the value of a file information item

CMS 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 a number that represents an information item, the value of which will be returned by the function.

Details

The following table defines the information that is returned for a given *info-item* value.

*info-item*Value	Information Items Available For...	
	Single Files	Concatenated Files
1	file name	file name
2	owner name	file list
3	group name	owner name
4	access permission	group name
5	file size (bytes)	access permission
6		file size (bytes)

FINFO Output for Disk Files

The following example and output illustrate the use of FINFO and the other file access functions for disk files:

```
data _null_;
   length opt $100 optval $100;
   rc=FILENAME('myfile', 'finfo list *'); /* allocate file   */
   fid = FOPEN('myfile');                 /* open file       */
   infocnt=FOPTNUM(fid);  /* get number of information items */

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

   rc = FCLOSE(fid);                      /* close file      */
   rc = FILENAME('myfile');               /* deallocate file */
run;
```

Output 14.3 FINFO Output for a Disk File

```
Information for a File:
File Name          FINFO LIST A
Lrecl              80
Recfm              F
Blksize            960
NOTE: DATA statement used:
      real time           0.86 seconds
      cpu time            0.27 seconds
```

FINFO Output for Sequential Files

The following example and output illustrate the use of FINFO and the other file access functions for sequential files:

```
data _null_;
   length opt $100 optval $100;
   rc=FILENAME('myfile', 'sltape sas .saspgms'); /* allocate file */
   fid = FOPEN('myfile');                        /* open file     */
   infocnt=FOPTNUM(fid);       /* get number of information items */

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

   rc = FCLOSE(fid);                       /* close file          */
   rc = FILENAME('myfile');                /* deallocate the file */
run;
```

Output 14.4 FINFO Output for a Sequential File

```
Information for a File:
File Name          SLTAPE SAS SFSFP:USER1.SASPGMS
Lrecl              80
Recfm              F
Blksize            960
NOTE: DATA statement used:
      real time           0.15 seconds
      cpu time            0.13 seconds
```

See Also

- □ "FCLOSE" on page 146
- □ "FOPEN" on page 153
- □ "FOPTNAME" on page 154
- □ "FOPTNUM" on page 155
- □ *SAS Language Reference: Dictionary*

FOPEN

Opens an external file and returns a file identifier value

CMS specifics: You must close FOPEN files with the FCLOSE function

Syntax

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

fileref
specifies the fileref that you assign to the external file. You can assign filerefs by using the FILENAME statement or the FILENAME function.

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 mode can be used for pipes and other sequential devices such as READER.
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:

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.
V	file contains variable length records.

Details

Under CMS, you must close files that you open with FOPEN using the FCLOSE function at the end of a DATA step; files are not closed automatically after processing.

See “FINFO” on page 150 for an example that demonstrates the use of the FOPEN, FCLOSE, FINFO, FILENAME, FOPTNAME and FOPTNUM functions.

See Also

- “FCLOSE” on page 146
- “FINFO” on page 150
- “FOPTNAME” on page 154
- “FOPTNUM” on page 155
- *SAS Language Reference: Dictionary*

FOPTNAME

Returns the name of an item of information about a file

CMS 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 name of the information item that will be returned by the function.

Details

See “FINFO” on page 150 for:

- valid values for *info-item*
- definitions of names returned by FOPTNAME
- an example showing the use of FOPTNAME and other file access functions.

See Also

- □ "FCLOSE" on page 146
- □ "FOPEN" on page 153
- □ "FOPTNUM" on page 155
- □ *SAS Language Reference: Dictionary*

FOPTNUM

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

CMS specifics: *info-item*

Syntax

FOPTNUM (*file-id*)

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

Details

The FOPTNUM function returns the number of information items that were associated with the *file-id* when the file was opened with the FOPEN function.

See "FINFO" on page 150 for definitions of *info-item* values and for an example showing the use of FOPTNUM and the other file access functions.

See Also

- □ "FCLOSE" on page 146
- □ "FOPEN" on page 153
- □ "FOPTNAME" on page 154
- □ *SAS Language Reference: Dictionary*

GETEXEC

Returns the value of an EXEC variable

CMS specifics: all

Syntax

GETEXEC ('*argument*')

'argument'
is the name of the EXEC variable. It must be uppercased and enclosed in quotes.

Details

The GETEXEC function is a SAS DATA step function that returns the value of an EXEC variable. The EXEC must be written in EXEC2 or REXX.

Here are some points to consider when you are using the GETEXEC function:

- All EXEC variable names that are specified as arguments for GETEXEC must be in uppercase. This is true for REXX as well as EXEC2, even though REXX makes no distinction between upper- and lowercase in its variable names. If you specify a REXX compound symbol, such as NAME.1, as an argument, only the stem (the part of the name preceding the period) must be in uppercase.
- Although EXEC2 variables always begin with an ampersand (&), you must omit the ampersand when you use the name as an argument.
- The CMS EXEC processor hides all variables except those in the current EXEC. Hidden variables are not accessible by GETEXEC.
- Arguments to GETEXEC must be character values, and the values of EXEC variables, even numeric values, are always retained in character format. By default, SAS converts values automatically from numeric to character or character to numeric. However, it is better to use the DATA step functions INPUT and PUT to convert character values from EXEC variables to a specific numeric format, and vice versa.
- If the interface to EXEC variables fails, GETEXEC assumes that its arguments are invalid (because no EXEC variables can be accessed), and a message is written to the SAS log.

If an EXEC2 variable name used as an argument to GETEXEC does not exist, the GETEXEC function returns a missing value. If a REXX variable name that was used as an argument to the GETEXEC function does not exist, the GETEXEC function returns the name of the variable. For either EXEC2 or REXX, if the variable has a null value or a value of all blanks, GETEXEC returns a missing value.

Example

The following statement assigns the value of the EXEC variable LABEL to the SAS variable DISKID:

```
diskid=getexec('LABEL');
```

See Also

- "Using the GETEXEC DATA Step Function" on page 97

KTRANSLATE

Replaces specific characters in a character expression

CMS specifics: to/from pairs

Syntax

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

Details

Under CMS, every *to* argument must have a corresponding *from* argument; you cannot specify a null *from* argument. There is no practical limit to the number of *to-from* pairs.

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 160
- *SAS Language Reference: Dictionary*

LIBNAME

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

CMS specifics: *libref* can also be a DDname name

Syntax

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

Details

Under CMS, DDnames assigned by using the CMS FILEDEF command can be used to refer to SAS data libraries. See "Using the CMS FILEDEF Command" on page 32 for more information.

The LIBNAME function accepts as arguments the same host-specific options that are available for the LIBNAME statement. The following example shows how the SEGMENT= option can be specified in a LIBNAME function:

```
data _null;
a=(libname('v8rdata',,,'segment=yes'));
```

For further information, see "LIBNAME" on page 243.

See Also

- *SAS Language Reference: Dictionary*

MOPEN

Opens a file by directory ID and member name and returns either the file identifier or a 0

CMS specifics: directory specification

Syntax

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

Details

You must open the directory referenced by the directory ID before you open a file using MOPEN. Under CMS, MOPEN can open files for output and for append, except for MACLIB members, which MOPEN can open for input only.

See Also

- "DOPEN" on page 144
- *SAS Language Reference: Dictionary*

PATHNAME

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

CMS specifics: *fileref*

Syntax

PATHNAME (*fileref*)

fileref
specifies the fileref that was assigned to an external file or to a SAS data library. Under CMS, *fileref* can also be a DDname that was assigned with a CMS FILEDEF command.

Details

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

See Also

- *SAS Language Reference: Dictionary*

RANK

Returns the position of a character in the EBCDIC collating sequence

CMS specifics: EBCDIC collating sequence

Syntax

RANK (*n*)

n
is a character in the EBCDIC collating sequence represented by a specific integer.

Details

Under CMS, the RANK function returns an integer that represents the position of a character in the EBCDIC collating sequence. Some operating environments return the ASCII equivalent.

Example

The following example returns a value of 193 and assigns it to the variable *n*:

```
n = rank('A');
```

See Also

- *SAS Language Reference: Dictionary*

SYSGET

Returns the value of the specified operating environment variable or symbol

CMS specifics: *operating-environment-variable, group*

Syntax

SYSGET ('<*group*>*operating-environment-variable*')

group
is an optional GLOBALV group containing the variable. If *group* is omitted, the default group is SAS.

operating-environment-variable
is the name of an operating environment variable or symbol that has been created with the GLOBALV command or with the SET= system option.

Details

Under CMS, *operating-environment-variable* is the name of an operating environment variable or symbol that has been created with the GLOBALV command.

See Also

- “Accessing System Variables” on page 10
- *SAS Language Reference: Dictionary*

SYSTEM

Invokes a CMS or CP command and returns the return code

Alias: CMS

Details

See “CMS” on page 140

TRANSLATE

Replaces specific characters in a character expression

CMS specifics: pairs of *to* and *from* arguments

Syntax

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

Details

Under CMS, every *to* argument must have a corresponding *from* argument; you cannot specify a null *from* argument. There is no practical limit to the number of *to* and *from* argument pairs.

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 156
- □ "SET=" on page 291
- □ *SAS Language Reference: Dictionary*

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

SAS® Companion for the CMS Environment, Version 8

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

ISBN 1–58025–481–0

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

1st printing, October 1999