SAS Functions under OpenVMS

A SAS function returns a value from a computation or system operation. Most functions use arguments that are supplied by the user as input. Most SAS functions are completely described in SAS Language Reference: Dictionary. The functions that are described here have syntax or behavior that is specific to the OpenVMS operating environment.

Using Terminal-Access Functions

In the following sections, a category is listed immediately following the name and short description of each function. Most of these categories are self-explanatory. For terminal-access functions, which enable you to get information from and write information to the terminal, please observe the following caution:

**CAUTION:**

Do not use the terminal-access functions in the windowing environment. Terminal-access functions work in the windowing environment, but they can either overwrite the display or be overwritten by the display. (The REFRESH (CTRL-R) command can be used to restore your display.) For details about the REFRESH command, see the SAS online Help. △

Under OpenVMS, the following SAS functions are terminal-access functions:

- SETTERM
- TERMIN
- TERMOUT
- TTCLOSE
- TTCONTRL
- TTCOPEN
- TTREAD
TTWRITE

SAS CALL Routines under OpenVMS

SAS CALL routines are used to alter variable values or perform other system functions. Most CALL routines are completely described in SAS Language Reference Dictionary. The CALL routines that are described here have syntax or behavior that is specific to the OpenVMS operating environment.

Dictionary

ASCEBC

Converts an input character string from ASCII to EBCDIC

Language element: function
Category: character-string translation
OpenVMS specifics: All aspects are host-specific

Syntax

ASCEBC (in-string)

in-string

is any ASCII string, and can be a character variable, a character literal enclosed in quotation marks, or another character expression. The value of in-string is limited to 200 characters.

Details

The return value is the EBCDIC translation of in-string.

BYTE

Returns one character in the ASCII collating sequence

Language element: function
Category: character
OpenVMS specifics: ASCII collating sequence
Functions and CALL Routines

**Syntax**

`BYTE(n)`

**Details**

`n` is an integer representing a specific ASCII character. Under OpenVMS, `n` can range from 0 to 255.

**See Also**

- BYTE function in SAS Language Reference Dictionary

---

**CALL FINDEND**

Releases resources that are associated with a directory search

**Language element:** CALL routine  
**Category:** general purpose OpenVMS  
**OpenVMS specifics:** All aspects are host-specific

**Syntax**

`CALL FINDEND(context)`

**Details**

`context` is the same as the context variable that is used by the FINDFILE function to maintain the search context between executions of FINDFILE. The context argument must be initialized before FINDFILE is called. Also, the value of context must not be manipulated before it is used in the CALL FINDEND routine; if it is, channels and resources cannot be freed to the process until the process terminates.

**Example**

In the following example, FINDFILE is used to search the user's directories for a filename that matches MYPROG*.SAS. If it finds a file named MYPROG12.SAS, for
example, then FN is set to *myprog12.sas*. The CALL FINDEND routine is then called to terminate the directory search and to release the associated resources.

```context=0;
fn=findfile("myprog*.sas",context);
do while (fn ^= " ");
   put fn;
   fn=findfile("myprog*.sas",context);
end;
call findend(context);
```

**See Also**

- Function: “FINDFILE” on page 289

---

**CALL SYSTEM**

**Issues operating environment commands**

**Language element:** CALL routine

**Category:** special

**OpenVMS specifics:** Issues DCL commands; some commands execute in a subprocess, others in the parent process

**Syntax**

```CALL SYSTEM(DCL-command)```

**DCL-command**

can be any of the following under OpenVMS:

- a DCL command enclosed in quotation marks
- an expression whose value is a DCL command
- the name of a character variable whose value is a DCL command.

**Details**

In the windowing environment, a new window is displayed when the command executes. Any output from the command is displayed (for example, a directory listing). Select the File menu and click on Exit to remove this window.

Note that some DCL commands execute in the parent OpenVMS process and some execute in a subprocess. For more information, see “Issuing DCL Commands during a SAS Session” on page 37.

**Comparisons**

The CALL SYSTEM routine is similar to the X statement, the X command, the %SYSEXEC macro, and the VMS function; however it can be called conditionally. In most cases, the X statement, the X command, or the %SYSEXEC macro 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. The benefit of the CALL SYSTEM routine being callable is that it is not executed unconditionally at DATA step compile time, whereas other methods are.

Example

The following is an example of the CALL SYSTEM routine:

```sas
data _null_
   call system('define mylib [mydir.datasets]');
run;
```

See Also

- CALL SYSTEM routine in SAS Language Reference: Dictionary
- “Issuing DCL Commands during a SAS Session” on page 37
- Command: “X” on page 235
- Function: “VMS” on page 318
- Statement: “X” on page 384
- %SYSGET macro in “Macro Functions” on page 464

---

**COLLATE**

Generates an ASCII collating sequence character string

Language element: function

Category: character

OpenVMS specifics: ASCII collating sequence

**Syntax**

```
COLLATE(start-position,<end-position>) | (start-position<,,length>)
```

**start-position**

specifies the ASCII character where the collating sequence is to begin.

**end-position**

specifies the ASCII character where the collating sequence is to end.

**length**

specifies the number of characters in the returned string.

**Details**

The COLLATE function returns a string of ASCII characters, which can range in value from 0 to 255. The string returned by the COLLATE function begins with the ASCII
character specified by start-position. (Characters 128 to 255 are usually special control characters such as special fonts, but the COLLATE function returns them.) If end-position is specified, the string returned by the COLLATE function contains all the ASCII characters between start-position and end-position. If length is specified instead of end-position, then the COLLATE function returns a string of length. The returned string ends, or truncates, with the character having the value 255 if you request a string length that contains characters exceeding this value. If you specify both end-position and length, the COLLATE function ignores length. If you request a string longer than the remainder of the sequence, COLLATE returns a string through the end of the sequence. Unless you assign the return value of the COLLATE function to a variable with a defined length of less than 200, the ASCII collating sequence string is padded with blanks to a length of 200. If you request more than 200 characters, the returned string is truncated to a length of 200.

See Also

- COLLATE function in SAS Language Reference: Dictionary

---

DELETE

Deletes a file

Language element: function

Category: general-purpose OpenVMS

OpenVMS specifics: All aspects are host-specific

Syntax

DELETE('filename')

'filename'

is the name of the file to be deleted. It can be a character variable, a character literal enclosed in quotation marks, or another character expression. The value for filename must be enclosed in quotation marks.

Details

If the DELETE function executes successfully, the return value is 0. Otherwise, the return value is any of the OpenVMS error codes that indicate why it failed. The following are two common error codes:

98962 File not found.

98970 Insufficient privilege or file protection violation.

The text of the error codes is retrieved using the GETMSG function.
See Also

- Function: “GETMSG” on page 296

DINFO

Returns information about a directory
Language element: function
Category: external-file
OpenVMS specifics: valid values for info-item; returned values

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 information item to be retrieved.

Details

The DINFO function returns the value of a system-dependent directory parameter.

See Also

- DINFO function in SAS Language Reference: Dictionary
- Function: “DOPEN” on page 279
- Function: “DOPTNAME” on page 280
- Function: “DOPTNUM” on page 281

DOPEN

Opens a directory and returns a directory identifier value
Language element: function
Category: external-file
OpenVMS specifics: valid values for fileref

Syntax

DOPEN('fileref')
‘fileref’

specifies the fileref assigned to the directory. The value for fileref must be enclosed in quotation marks.

Details

The DOPEN function opens a directory and returns a directory identifier value (a number greater than 0) that is used to identify the open directory in other SAS external file access functions. If the directory could not be opened, DOPEN returns a value of 0. The directory to be opened must be identified by a fileref.

See Also

- DOPEN function in SAS Language Reference: Dictionary
- Function: “DINFO” on page 279
- Function: “DOPTNAME” on page 280
- Function: “DOPTNUM” on page 281

---

**DOPTNAME**

*Returns directory attribute information*

Language element: function

Category: external-file

OpenVMS specifics: valid values for nval; number of options available

---

**Syntax**

**DOPTNAME**(directory-id,nval)

directory-id

specifies the identifier that was assigned when the directory was opened, generally by the DOPEN function.

Restriction: To use DOPTNAME on a directory, the directory must have been previously opened by using the DOPEN function.

nval

specifies the sequence number of the directory-information item (as listed by the DOPTNUM function).

**Details**

The number, names, and nature of the directory information varies between operating environments. The number of options that is available for a directory varies depending on the operating environment.
See Also

- DOPTNAME function in SAS Language Reference: Dictionary
- Function: “DINFO” on page 279
- Function: “DOPEN” on page 279
- Function: “DOPTNUM” on page 281

DOPTNUM

Returns the number of information items available for a directory

Language element: function
Category: external-file
OpenVMS specifics: valid values for directory-id; number of options available

Syntax

DOPTNUM(directory-id)

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

Details

The number, names, and nature of the directory information vary between operating environments. The number of options that are available for a directory varies depending on the operating environment. The directory specified by directory-id must have been previously opened by using the DOPEN function.

See Also

- DOPTNUM function in SAS Language Reference: Dictionary
- Function: “DINFO” on page 279
- Function: “DOPEN” on page 279
- Function: “DOPTNAME” on page 280

EBCASC

Converts an input character string from EBCDIC to ASCII

Language element: function
Category: character-string translation
OpenVMS specifics: All aspects are host-specific

Syntax

\texttt{EBCASC(in-string)}

\textbf{in-string}

is any EBCDIC string, and can be a character variable, a character literal enclosed in quotation marks, or another character expression. The value of \texttt{in-string} is limited to 200 characters.

Details

The return value is the ASCII translation of \texttt{in-string}.

\section*{FDELETE}

Deletes an external file or an empty directory

Language element: function

Category: external-file

OpenVMS specifics: valid values for directory

Syntax

\texttt{FDELETE('fileref' | directory)}

\textbf{fileref}

specifies the fileref that you assign to the external file. The value for fileref must be enclosed in quotation marks.

\textbf{directory}

specifies an empty directory that you want to delete.

Details

The FDELETE function allows you to delete an external file or an empty directory. Under OpenVMS, filerefs can be assigned by environment variables and by system commands.
See Also

- FDELETE function in SAS Language Reference: Dictionary

FEXIST

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

Language element: function
Category: external-file
OpenVMS specifics: valid values for fileref

Syntax

FEXIST("fileref")

“fileref”

specifies the fileref assigned to an external file. The fileref must have been previously assigned. The value for fileref must be enclosed in either single or double quotation marks.

Details

The FEXIST function returns a value of 1 if the external file that is associated with fileref exists, and a value of 0 if the file does not exist. You can assign filerefs by using the FILENAME statement or the FILENAME function.

See Also

- FEXIST function in SAS Language Reference: Dictionary
- Statement: “FILENAME” on page 359
- Function: “FILENAME” on page 285

FILEATTR

Returns the attribute information for a specified file

Language element: function
Category: general-purpose OpenVMS
OpenVMS specifics: All aspects are host-specific

Syntax

FILEATTR(file-specification,item)
**file-specification**

is the file for which you are requesting information. It can be a character variable, a character literal enclosed in quotation marks, or another character expression. You must have access to the file that you are referencing.

**item**

specifies which attribute of the file you are requesting. It can be a character variable, a character literal enclosed in quotation marks, or another character expression. If the item is more than 16 characters long, it is truncated. The items that can be requested are the same as the items that can be requested using the DCL lexical function F$FILE_ATTRIBUTE.

**Details**

The FILEATTR function returns information about a file based on the type of information that is requested with the item parameter. Numeric values are returned as character values.

The FILEATTR function closely resembles the F$FILE_ATTRIBUTE lexical function of DCL. For more information about DCL lexical functions, refer to OpenVMS DCL Dictionary or to the SAS online Help.

You cannot request the following attribute information:

- after-image journaling (AI)
- before-image journaling (BI)
- recovery-unit journaling (RU).

---

**FILEEXIST**

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

Language element: function
Category: external-file
OpenVMS specifics: valid values for file-name

**Syntax**

FILEEXIST(“file-name”)

“file-name”

specifies a fully qualified physical filename of an OpenVMS file. The value for file-name must be enclosed in either single or double quotation marks.

**Details**

The FILEEXIST function returns a value of 1 if the external file exists and a value of 0 if the external file does not exist.

You must always use fully qualified physical names with the FILEEXIST function.
FILENAME

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

Language element: function
Category: external-file
OpenVMS specifics: valid values for file-name, device, and dir-ref

Syntax
FILENAME(fileref,file-name
    <,device,host-options,<dir-ref>>>)

fileref
    in a DATA step, specifies the fileref to assign to an external file. (For details, see the FILENAME function in SAS Language Reference Dictionary.)

file-name
    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:
    DISK
        specifies a disk.
    DUMMY
        specifies that the output to the file is discarded.
    GTERM
        specifies the graphics on the user's terminal.
    PLOTTER
        specifies an unbuffered graphics output device.
    PRINTER
        specifies a printer or printer spool file.
    TAPE
        specifies a tape driver or tape device.
TEMP
specifies a temporary file that can only be accessed through the logical name and
is only available while the logical name exists. If a physical pathname is specified,
an error is returned. Files manipulated by the TEMP device can have the same
attributes and behave identically to DISK files.

TERMINAL
specifies the user's terminal.

host-options
can be any of the following:

ALQ=
specifies how many disk blocks to allocate to a new external file. The value can
range from 0 to 2,147,483,647. If the value is 0 (the default), the minimum
number of blocks required for the given file format is used.

CC=
tells SAS what type of carriage control to use when it writes to external files.
Values for the CC= option are

FORTRAN indicates FORTRAN carriage-control format. This is the
default for print files.

PRINT indicates OpenVMS print format.

CR indicates OpenVMS carriage-return carriage-control format.
This is the default for nonprinting files.

DEQ=
tells OpenVMS how many disk blocks to add when it automatically extends an
external file during a write operation. The value can range from 0 to 65,535. The
default value is 0, telling OpenVMS RMS to use the process's default value.

FAC=
overrides the default file access attributes used for external files. Values for the
FAC= option are

DEL specifies delete access.

GET specifies read access.

PUT specifies write access.

UPD specifies update access.

GSFCC=
specifies the file format of graphic stream files (GSF files). The accepted values are

PRINT creates a GSF file. It is a VFC format file with carriage control
set to null. These files can be used with most utilities with the
exception of some file transfer protocols, such as Kermit. This
is the default value for this option.

CR creates a carriage return carriage control file.

NONE creates a file with no carriage control. This format is useful if
you plan to download the file to a personal computer.
KEY=

specifies which key the SAS System uses to read the records in an RMS file with indexed organization. The KEY= option is always used with the KEYVALUE= option.

KEYVALUE=

specifies the key value with which to begin reading an indexed file.

LINESIZE=

specifies the line size for input or output. The value can range from 10 to 32,768. The default is 80 for interactive jobs (interactive line mode and the SAS windowing environment) and 132 for noninteractive and batch jobs for print files.

LRECL=

specifies the record length of the output file. If you do not specify a record length, the default is varying length records. For input, the existing record length is used by default. If the LRECL= option is used, the input records are padded or truncated to the specified length.

The maximum LRECL= value for the OpenVMS operating environment is 65535; however, errors can be produced if you attempt to access certain file types with an LRECL of less than 65535.

MBC=

specifies the size of the I/O buffers that OpenVMS RMS allocates for a particular file. The value can range from 0 to 127 and represents the number of blocks used for each buffer. By default, this option is set to 0 and the default values for the process are used.

MBF=

specifies the number of I/O buffers you want OpenVMS RMS to allocate for a particular file. The value can range form 0 to 127 and represents the number of buffers used. By default, this option is set to 2 buffers. If a value of 0 is specified, the default value for the process is used.

MOD

opens the file referenced for append. This option does not take a value.

NEW

opens a new file for output. This option does not take a value.

OLD

opens a new file for output. This option does not take a value.

PAGESIZE=

specifies the page size for output. The default is the display setting for interactive jobs (interactive line mode and the SAS windowing environment) and 60 for noninteractive and batch jobs. The value can range from 20 to 500.

RECFM=

specifies the record format of the output file. Values for the RECFM= option are

F specifies fixed length.

V specifies variable length.
D specifies you are accessing unlabeled tapes with the PUT and INPUT DATA step statements. For more information, see “Reading from an Unlabeled Tape” on page 178.

SHR= overrides the default file-sharing attributes used for external files. Values for the SHR= option are

- DEL specifies delete access.
- GET specifies shared read access.
- NONE specifies no shared access.
- PUT specifies shared write access.
- UPD specifies update access.

You can combine these values in any order. For additional details about these options, see the discussion of host-specific external I/O statement options for the FILENAME statement in “FILENAME” on page 359.

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

**Details**

Under OpenVMS, you can assign filerefs using two methods. You can use the DCL DEFINE command to assign a fileref before you invoke SAS. For example:

```
$ define myfile a.txt
$ sas;
  data;
  file myfile;
  put 'HELLO';
  run;
```

This creates the file A.TXT.

You can use the X command to assign a fileref during your SAS session.

**See Also**

- FILENAME function in SAS Language Reference: Dictionary
- Function: “FILEREF” on page 288
- Statement: “FILENAME” on page 359
- Data set option: “ALQ=” on page 247
- Data set option: “DEQ=” on page 254
- System option: “CC=” on page 403
- Command: “X” on page 235

**FILEREF**

Verifies that a fileref has been assigned for the current SAS session and returns a value.
Language element: function
Category: external-file
OpenVMS specifics: valid values for fileref

Syntax
FILEREF(fileref)

fileref
specifies the fileref to be validated. Under OpenVMS, fileref can also be an OpenVMS logical name that was assigned using the DCL DEFINE command.

Details
Under OpenVMS, you can assign filerefs using two methods. You can use the DCL DEFINE command to assign a fileref before you invoke SAS. For example:

$ define myfile a.txt
$ sas;
  data;
  file myfile;
  put ’’HELLO’’;
  run;
This creates the file A.TXT.
  You can use the X command to assign a fileref during your SAS session.

See Also
- FILEREF function in SAS Language Reference Dictionary
- Function: “FILENAME” on page 285
- Command: “X” on page 235

FINDFILE

Searches a directory for a file
Language element: function
Category: general-purpose OpenVMS
OpenVMS specifics: All aspects are host-specific

Syntax
FINDFILE(file-specification,context)
**file-specification**

specifies the file specification of the file that you are searching for. It can contain any valid OpenVMS file specification, including wildcards. The value for file-specification can be a character variable, a character literal enclosed in quotation marks, or another character expression. You must have access to the file that you are searching for.

**context**

is a variable used internally by the SAS System to maintain the search context between executions of FINDFILE. It must be initialized to 0 before the first execution of FINDFILE for a given file-specification and must not be modified between executions. Context must be a numeric variable initialized to 0; it cannot be a literal 0. You can use FINDFILE for multiple search streams by specifying a different context variable for each stream. For example, you can have variables named CONTEXT1 and CONTEXT2.

### Details

The FINDFILE function searches all directories and subdirectories for file-specification and returns the first filename that matches the file specification given. Subsequent calls return other filenames that match the specification. For more information, see the description of the CALL FINDEND routine in “CALL FINDEND” on page 275.

The return value is the name of the file that matches file-specification. If no file matches or if the last one in the list has already been returned, a blank is returned. The target variable must be long enough to contain an OpenVMS pathname, which can be up to 255 characters long. SAS character variables have a maximum length of 32K or 32767.

### Example

The following example uses the FINDFILE function:

```sas
context=0;
fn=findfile('myprog*.sas',context);
do while (fn ^= ' ');
   put fn;
   fn=findfile('myprog*.sas',context);
end;
```

This example searches the user's directories for a filename that matches MYPROG*.SAS; for example, if it finds a file named MYPROG12.SAS, then FN is set to myprog12.sas.

### See Also

- CALL routine: “CALL FINDEND” on page 275

---

**FINFO**

Returns the value of a file information item

**Language element:** function
Category: external-file
OpenVMS specifics: types of file information

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 name of the file information item to be retrieved.

Details

The `FINFO` function returns the value of a system-dependent information item for an external file. `FINFO` returns a blank if the value given for `info-item` is invalid.

See Also

- `FINFO` function in SAS Language Reference Dictionary
- Function: “FOPEN” on page 291
- Function: “FOPTNAME” on page 292
- Function: “FOPTNUM” on page 292

---

**FOPEN**

Opens an external file and returns a file identifier value

Language element: function  
Category: external-file  
OpenVMS specifics: Files are not closed automatically after processing

Syntax

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

Note: This is a simplified version of the `FOPEN` function syntax. For the complete syntax and its explanation, see the `FOPEN` function in SAS Language Reference Dictionary.

`'fileref'`

specifies the fileref assigned to an external file. The value for `fileref` must be enclosed in quotation marks.
Details

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

See Also

- FOPEN function in SAS Language Reference: Dictionary
- Function: “FILENAME” on page 285
- Function: “FILEREF” on page 288

---

## FOPTNAME

Returns the name of an item of information about a file

Language element: function
Category: external-file
OpenVMS specifics: available information items

### Syntax

```sas
FOPTNAME(file-id,nval)
```

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

**nval**
- specifies the number of the information item.

### Details

The FOPTNAME function returns a blank if an error occurred.

### See Also

- FOPTNAME function in SAS Language Reference: Dictionary
- Function: “FILENAME” on page 285
- Function: “FOPEN” on page 291
- Function: “FOPTNUM” on page 292

---

## FOPTNUM

Returns the number of information items available about a file
Functions and CALL Routines

Language element: function
Category: external-file
OpenVMS specifics: available information items

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 available about a file.

See Also

- **FOPTNUM** function in SAS Language Reference: Dictionary
- Function: “FINFO” on page 290
- Function: “FOPEN” on page 291
- Function: “FOPTNAME” on page 292

---

GETDVI

Returns a specified item of information from a device

Language element: function
Category: general-purpose OpenVMS
OpenVMS specifics: All aspects are host-specific

Syntax

**GETDVI** (device-name, item)

**device-name**

specifies a physical device name or a logical name equated to a physical device name. Specify the device name as a character-string expression.

After the device-name argument is evaluated, the F$GETDVI lexical function examines the first character of the name. If the first character is an underscore (_), the name is considered a physical device name. Otherwise, a single level of logical name translation is performed, and the equivalence name, if any, is used.
**GETJPI**

Retrieves job-process information

Language element: function
Category: general-purpose OpenVMS
OpenVMS specifics: All aspects are host-specific

**Syntax**

`GETJPI(jpi-item, <pid>)`

**jpi-item**

is a character variable that contains any item accepted by the F$GETJPI lexical function, for example, a user process name. For more information about the F$GETJPI lexical function, see OpenVMS DCL Dictionary.

**pid**

can be either character (process-name variable) or numeric (process-ID variable). If the PID parameter is a character variable, GETJPI looks up information for a process whose name matches the value of the character variable. However, because of the way in which character variables are passed to functions, the GETJPI function must trim trailing blanks from the character variable. For this reason, you cannot use character variables to specify a process name if the process name itself contains trailing blanks. Instead, you should either use a numeric value to specify the process ID, or you should omit the trailing blanks from the name of the desired process. If you do not specify this argument, the current process is used.

**Details**

The GETJPI function returns the job-process information as a character string. If the job-process information string is longer than the length of the target variable, it is truncated.

**GETLOG**

Returns information about a DCL logical name
Language element: function
Category: general-purpose OpenVMS
OpenVMS specifics: All aspects are host-specific

Syntax
GETLOG (logical-name,<table>,<index>,
   <mode>,<case>,<item>)

logical-name
  can be a character variable, character literal enclosed in quotation marks, or another character expression. This required argument is the DCL logical name that you want information about.

table
  is an optional character parameter that is the name of a DCL logical name table. It can be a character variable, a character literal enclosed in double quotation marks, or another character expression. The default is “LNM$DCL_LOGICAL”. If the table name is more than 31 characters long, it is truncated. If table is specified, the GETLOG function searches only the specified table for the logical name.

  If you specify “CASE_SENSITIVE” in the case argument, then you must use the proper case in the table argument as well.

index
  is an optional numeric parameter that indicates the number of the translation to return if a logical name has multiple translations. This argument can be either a numeric literal or numeric variable. The default value is 0.

mode
  is an optional character parameter that contains the access mode to be used for translation. It can be a character variable, a character literal enclosed in double quotation marks, or another character expression. The default is “USER”. If the mode name is more than 10 characters long, it is truncated. If mode is specified, the GETLOG function searches only for a logical name created with the specified access mode.

case
  is an optional character parameter that determines the case to be used for translation. It can be a character variable, a character literal enclosed in double quotation marks, or another character expression. If the case name is more than 14 characters long, it is truncated.

  “CASE_BLIND”
   specifies to ignore the case of the characters for translation. This is the default.

  “CASE_SENSITIVE”
   specifies to accept the case of the characters for translation.

   If you specify “CASE_SENSITIVE” as the value for the case argument, then you must also use the correct case in the table argument value.
item

is an optional character parameter that specifies what type of information is to be
returned about a logical name. It can be a character variable, a character literal
enclosed in double quotation marks, or another character expression. The default
value is “VALUE”. If item is more than 11 characters long, it is truncated.

Details

The GETLOG function returns information about a DCL logical name. The return
string is always a character value. Numeric values are returned as character values.
The default return value is the equivalence name of a logical name.

The GETLOG function closely resembles the F$TRNLNM lexical function of DCL.
For more information about the syntax and arguments of the GETLOG function, such
as all valid values for a particular argument, refer to the F$TRNLNM lexical function
in OpenVMS DCL Dictionary or in the SAS online Help.

Note: You cannot skip any arguments when using the GETLOG function. For
example, in order to specify a value for item, you must also specify values for table,
index, mode, and case. If you do not want to change the values for these arguments,
then simply specify the default value.

GETMSG

Translates an OpenVMS error code into text

Language element: function
Category: general-purpose OpenVMS
OpenVMS specifics: All aspects are host-specific

Syntax

GETMSG(status)

status

is an OpenVMS status code. It is usually returned from one of the other functions
that return an OpenVMS status code on failure.

Details

The return value is a character variable that receives the message text corresponding to
the status code. If the message string is longer than the length of the target variable, it
is truncated.
GETQUOTA

Retrieves disk quota information

Language element: function
Category: general-purpose OpenVMS
OpenVMS specifics: All aspects are host-specific

Syntax
GETQUOTA(dev, user, usage, perm, over, context)

**dev**
is the device that you want to gather disk quota information for.

**user**
receives your numeric user identification code (UIC) on the disk. The UIC format can be used to format the numeric value. This variable must be initialized to 0 before the first execution.

**usage**
receives your current disk usage in blocks. This variable must be initialized to 0 before the first execution.

**perm**
receives your permanent quota. This variable must be initialized to 0 before the first execution.

**over**
receives your allowed overdraft. This variable must be initialized to 0 before the first execution.

**context**
is a numeric variable that must be initialized to 0 before the first execution and must not be modified between calls.

Details
Besides storing the quota information in the USER, USAGE, PERM, and OVER variables, the GETQUOTA function also returns the OpenVMS status code that is returned by SYS$QIO. The OpenVMS status code can have the following return codes:

1  indicates the GETQUOTA function was successful and more disk quota remains.
996  indicates that no more quota information is available.

See Also

- Function: “DELETE” on page 278
indicates that quotas are not enabled on the volume.

Any other value indicates an OpenVMS error.

Note: In order to use the GETQUOTA function, you must have either SYSPRV privileges or read access to QUOTA.SYS on the volume.

Example

The following example uses the GETQUOTA function:

data gquota;
    dev="$1$DUA0:"
    user=0;
    usage=0;
    perm=0;
    over=0;
    context=0;
    do until (rc ^= 1);
        rc=getquota(dev,user,usage,perm,over,context);
        output;
    end;
run;

    proc print data=gquota;
        run;

See Also

- Format: "UICw." on page 268

GETSYM

Returns the value of a DCL symbol

Language element: function

Category: general-purpose OpenVMS

OpenVMS specifics: All aspects are host-specific

Syntax

GETSYM(symbol-name)

symbol-name

is the name of a DCL symbol defined in your process. It can be a character variable, character literal enclosed in quotation marks, or another character expression. If symbol-name is more than 200 characters long, it is truncated.
Details
The return value is the character string equivalent of the DCL symbol. If the symbol is defined as both a local and global symbol, then the local value is returned. If the symbol value string is longer than the length of the target variable, it is truncated.

See Also

- Function: “SYSGET” on page 309

GETTERM

Returns the characteristics of your terminal device

Language element: function
Category: general-purpose OpenVMS
OpenVMS specifics: All aspects are host-specific

Syntax

GETTERM(characteristic-name)

characteristic-name

is the name of the terminal characteristic to be returned. The argument can be a character variable, character literal enclosed in quotation marks, or another character expression. If characteristic-name is longer than 200 characters, it is truncated.

Details

The GETTERM function returns the characteristics of your terminal device from within the SAS System. It can be called from either the DATA step or an SCL program. This function eliminates the need to use the X command or statement to return your terminal characteristics. The return value is a numeric code, which is the current setting of a characteristic.

Characteristic values that are Boolean (on or off) are returned as 0 or 1. Characteristic values that have integer values, such as page size, are returned as the function value.

If an error occurs during the execution of the function, GETTERM returns a negative result. Some common error return codes include the following:

-20 represents the OpenVMS symbolic name SS$_BADPARAM, which means the characteristic name is not valid or was specified ambiguously.

-2313 represents the OpenVMS symbolic name, SS$_NOSUCHDEV, which means the current SYS$OUTPUT device is not a terminal device, or does not exist.

Table 14.1 on page 300 lists in alphabetic order the characteristics that can be returned by the GETTERM function.
Table 14.1 Terminal Characteristics

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALTTYPEAHEAD</td>
<td>Alternate typeahead buffer enabled</td>
</tr>
<tr>
<td>ANSI CRT</td>
<td>Device is an ANSI CRT</td>
</tr>
<tr>
<td>APPLICATION</td>
<td>Keypad is in application mode</td>
</tr>
<tr>
<td>AUTOBAUD</td>
<td>Automatic baud rate detection is enabled</td>
</tr>
<tr>
<td>AVO</td>
<td>Terminal has advanced video option</td>
</tr>
<tr>
<td>BLOCK</td>
<td>Terminal is in block transfer mode</td>
</tr>
<tr>
<td>BROADCAST</td>
<td>Terminal accepts broadcast messages</td>
</tr>
<tr>
<td>BROADCASTMBX</td>
<td>Broadcast messages sent via mailbox</td>
</tr>
<tr>
<td>DECCRT</td>
<td>Terminal is a DEC CRT (VT100 or later)</td>
</tr>
<tr>
<td>DECCRT2</td>
<td>Terminal is a DEC CRT (VT200 or later)</td>
</tr>
<tr>
<td>DIALUP</td>
<td>Terminal is on a dialup line</td>
</tr>
<tr>
<td>DISCONNECT</td>
<td>Terminal disconnects when hangup occurs</td>
</tr>
<tr>
<td>DMA</td>
<td>Terminal uses asynchronous DMA</td>
</tr>
<tr>
<td>DRCS</td>
<td>Terminal has soft character font set</td>
</tr>
<tr>
<td>ECHO</td>
<td>Terminal input is echoed</td>
</tr>
<tr>
<td>EDIT</td>
<td>Terminal has editing capabilities</td>
</tr>
<tr>
<td>EDITING</td>
<td>Terminal line editing is enabled</td>
</tr>
<tr>
<td>EIGHTBIT</td>
<td>Terminal accepts 8-bit escape codes</td>
</tr>
<tr>
<td>ESCAPE</td>
<td>Terminal validates escape sequences</td>
</tr>
<tr>
<td>FALLBACK</td>
<td>Output is transformed by TFF</td>
</tr>
<tr>
<td>FORMFEED</td>
<td>Terminal has mechanical form feed</td>
</tr>
<tr>
<td>HALFDUPLEX</td>
<td>Terminal is in half-duplex mode</td>
</tr>
<tr>
<td>HANGUP</td>
<td>Modem is hung up when terminal logs out</td>
</tr>
<tr>
<td>HOSTSYNC</td>
<td>Host system is synchronized to terminal</td>
</tr>
<tr>
<td>INSERT</td>
<td>Default mode is insert instead of overstrike</td>
</tr>
<tr>
<td>LINESIZE</td>
<td>Sets terminal line size</td>
</tr>
<tr>
<td>LOCALECHO</td>
<td>Command line read operations are echoed</td>
</tr>
<tr>
<td>LOWER</td>
<td>Terminal accepts lowercase characters</td>
</tr>
<tr>
<td>MAILBOX</td>
<td>Terminal does not use associated mailbox</td>
</tr>
<tr>
<td>MODEM</td>
<td>Terminal is connect via a modem</td>
</tr>
<tr>
<td>MODHANGUP</td>
<td>Modify hangup behavior</td>
</tr>
<tr>
<td>PAGESIZE</td>
<td>Sets terminal page size</td>
</tr>
<tr>
<td>PASSTHROUGH</td>
<td>Pass all characters unmodified/examined</td>
</tr>
<tr>
<td>PRINTER</td>
<td>Device has a printer port</td>
</tr>
<tr>
<td>READSYNC</td>
<td>Read synchronization is enabled</td>
</tr>
</tbody>
</table>
LIBNAME

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

Language element:  function
Category:  SAS file I/O
OpenVMS specifics:  valid values for SAS-data-library

Syntax

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

SAS-data-library

is the name of the directory that contains the SAS data library, enclosed in quotation marks. You can omit this argument if you are merely specifying the engine for a libref or an OpenVMS logical name that you previously assigned.

If the directory that you specify does not already exist, then you must create it before you attempt to use the libref that you have assigned to it.

Details

The SAS-data-library has a value of ‘’[]’’ (with no space) to assign a libref to the current directory. (The behavior of the LIBNAME function when a single space is specified for the SAS data library is host dependent.) If no value is provided for SAS-data-library or if SAS-data-library has a value of ‘’’’ (with no space), the LIBNAME function dissociates the libref from the data library.
Under OpenVMS, OpenVMS logical names (assigned by using the DCL DEFINE command) can also be used to refer to SAS data libraries. For more information, see “Assigning OpenVMS Logical Names” on page 129.

See Also

- LIBNAME function in SAS Language Reference: Dictionary
- “Assigning Librefs” on page 126
- Statement: “LIBNAME” on page 380

---

**LIBREF**

Verifies that a libref has been assigned and returns a value

Language element: function

Category: SAS file I/O

OpenVMS specifics: syntax

**Syntax**

LIBREF("libref")

"libref" specifies the libref to be verified. The value for libref must be enclosed in either single or double quotation marks.

**Details**

The LIBREF function returns a value of 0 if the operation was successful and a value of ≠0 if it was not successful.

See Also

- LIBREF function in SAS Language Reference: Dictionary
- Function: “LIBNAME” on page 301

---

**MOPEN**

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

Language element: function

Category: external-file

OpenVMS specifics: valid values for directory-id
Functions and CALL Routines

NODENAME 303

Syntax

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

Note: This is a simplified version of the MOPEN function syntax. For the complete syntax and its explanation, see the MOPEN function in SAS Language Reference: Dictionary.

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

Details

The MOPEN function returns the identifier for the file, or 0 if the file could not be opened.

See Also

- MOPEN function in SAS Language Reference: Dictionary
- Function: “DOPEN” on page 279

NODENAME

Returns the name of the current node

Language element: function
Category: general-purpose OpenVMS
OpenVMS specifics: All aspects are host-specific

Syntax

NODENAME()

Details

This function takes no arguments. The returned value can be up to 16 bytes long. In the following example, executing the statement on a node with node name of MYVAX assigns the value MYVAX to the variable THISNODE:

data _null_
   thisnode=nodename();
run;
PATHNAME

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

Language element: function
Category: SAS file I/O
OpenVMS specifics: fileref can be an OpenVMS logical name

Syntax

PATHNAME('fileref')

'fileref'
specifies the fileref that was assigned to an external file or to a SAS data library. Under OpenVMS, fileref can also be an OpenVMS logical name that was assigned with a DCL DEFINE command. The value of fileref must be enclosed in quotation marks.

Details

The PATHNAME function returns the physical name of an external file or SAS library, or blank if fileref or libref is invalid. The default length of the target variable in the DATA step is 200 characters.

See Also

- PATHNAME function in SAS Language Reference: Dictionary

PUTLOG

Creates an OpenVMS logical-name in your process-level logical name table

Language element: function
Category: general-purpose OpenVMS
OpenVMS specifics: All aspects are host-specific

Syntax

RC = PUTLOG(logical-name,value)

RC

is the return code. If the PUTLOG function executes successfully, the return value is 0. Otherwise, the return value is the OpenVMS error code indicating why it failed.
**logical-name**
the name of the OpenVMS logical name that you want to create. It can be a character variable, a character literal enclosed in quotation marks, or another character expression.

**value**
is the string to be assigned to the symbol. It can be a character variable, a character literal enclosed in quotation marks, or another character expression.

**Details**
The PUTLOG function creates an OpenVMS logical name in your process-level logical name table.

---

**PUTSYM**

**Syntax**
RC = PUTSYM(symbol-name, value, scope)

**RC**
is the return code. If the PUTSYM function executes successfully, the return value is 0. Otherwise, the return value is the OpenVMS error code indicating why it failed.

**symbol-name**
is the name of the DCL symbol that you want to create. It can be a character variable value, a character literal enclosed in single or double quotation marks, or another character expression.

**value**
is the string to be assigned to the symbol. It can be a character variable, a character literal enclosed in single or double quotation marks, or another character expression.

**scope**
defines whether the symbol is a local or global symbol. If the value of scope is 1, the symbol is defined as a local symbol. If the value of scope is 2, the symbol is defined as a global symbol. The scope argument can be either a numeric literal or a numeric variable.

**Details**
The PUTSYM function creates a DCL symbol in your process. If the PUTSYM function executes successfully, the return value is 0. Otherwise, the return value is the OpenVMS error code that indicates why it failed.
**RANK**

Returns the position of a character in the ASCII collating sequence

Language element: function  
Category: character  
OpenVMS specifics: ASCII collating sequence

**Syntax**

\[ \text{RANK}(x) \]

\( x \)

represents a character in the ASCII collating sequence.

**Details**

Because OpenVMS is an ASCII system, the RANK function returns an integer representing the position of a character in the ASCII collating sequence. If the length of \( x \) is greater than 1, you receive the rank of the first character in the string.

**See Also**

- RANK function in SAS Language Reference Dictionary

---

**RENAME**

Renames a file

Language element: function  
Category: general-purpose OpenVMS  
OpenVMS specifics: All aspects are host-specific

**Syntax**

\[ \text{RENAME}(\text{old-name},\text{new-name}) \]

\( \text{old-name} \)

is the current name of the file. It can be a character variable, a character literal enclosed in single or double quotation marks, or another character expression.

\( \text{new-name} \)

is the new name of the file. It can be a character variable, character literal enclosed in single or double quotation marks, or another character expression.
Details

You must have proper access to the file. If the RENAME function executes successfully, the return value is 0. Otherwise, the return value is the OpenVMS error code that indicates why it failed.

The following are two common error codes:

98962 File not found.
98970 Insufficient privilege or file protection violation.

The text of the error codes is retrieved using the GETMSG function.

See Also

- Function: “GETMSG” on page 296

---

SETTERM

Modifies a characteristic of your terminal device

Language element: function
Category: terminal-access
OpenVMS specifics: All aspects are host-specific

Syntax

SETTERM(characteristic-name,new-value)

characteristic-name

is the name of the terminal characteristic to be modified. The argument can be a character variable, a character literal enclosed in double quotation marks, or another character expression.

new-value

is the new setting for the characteristic.

Details

The SETTERM function modifies a terminal characteristic from within the SAS System. The SETTERM function can be called from either the DATA step or an SCL program. This function eliminates the need to use an X command or statement to modify your terminal characteristics.

The return value is a numeric status code, which is the previous setting of the characteristic, before the characteristic is changed by the function call.

Characteristic values that are Boolean (on or off) are returned as 1 or 0. Characteristic values that have integer values, such as page size, are returned as the function value.

If an error occurs during the execution of the SETTERM function, the result returned is negative. Some common error return codes include the following:
-20 represents the OpenVMS symbolic name SS$BADPARAM, which means that the characteristic name is not valid or was specified ambiguously.

-2313 represents the OpenVMS symbolic name, SS$NOSUCHDEV, which means the current SYS$OUTPUT device is not a terminal device, or does not exist.

The characteristics that can be set with the SETTERM function are the same as those that can be returned by the GETTERM function, and they are listed in Table 14.2 on page 308.

Table 14.2  Terminal Characteristics

<table>
<thead>
<tr>
<th>Characteristic</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>ALTTYPEAHEAD</td>
<td>Alternate typeahead buffer enabled</td>
</tr>
<tr>
<td>ANSICRT</td>
<td>Device is an ANSI CRT</td>
</tr>
<tr>
<td>APPLICATION</td>
<td>Keypad is in application mode</td>
</tr>
<tr>
<td>AUTOBAUD</td>
<td>Automatic baud rate detection is enabled</td>
</tr>
<tr>
<td>AVO</td>
<td>Terminal has advanced video option</td>
</tr>
<tr>
<td>BLOCK</td>
<td>Terminal is in block transfer mode</td>
</tr>
<tr>
<td>BROADCAST</td>
<td>Terminal accepts broadcast messages</td>
</tr>
<tr>
<td>BROADCASTMBX</td>
<td>Broadcast messages sent via mailbox</td>
</tr>
<tr>
<td>DECCRT</td>
<td>Terminal is a DEC CRT (VT100 or later)</td>
</tr>
<tr>
<td>DECCRT2</td>
<td>Terminal is a DEC CRT (VT200 or later)</td>
</tr>
<tr>
<td>DIALUP</td>
<td>Terminal is on a dialup line</td>
</tr>
<tr>
<td>DISCONNECT</td>
<td>Terminal disconnects when hangup occurs</td>
</tr>
<tr>
<td>DMA</td>
<td>Terminal uses asynchronous DMA</td>
</tr>
<tr>
<td>DRCS</td>
<td>Terminal has soft character font set</td>
</tr>
<tr>
<td>ECHO</td>
<td>Terminal input is echoed</td>
</tr>
<tr>
<td>EDIT</td>
<td>Terminal has editing capabilities</td>
</tr>
<tr>
<td>EDITING</td>
<td>Terminal line editing is enabled</td>
</tr>
<tr>
<td>EIGHTBIT</td>
<td>Terminal accepts 8-bit escape codes</td>
</tr>
<tr>
<td>ESCAPE</td>
<td>Terminal validates escape sequences</td>
</tr>
<tr>
<td>FALLBACK</td>
<td>Output is transformed by TFF</td>
</tr>
<tr>
<td>FORMFEED</td>
<td>Terminal has mechanical form feed</td>
</tr>
<tr>
<td>HALF Duplex</td>
<td>Terminal is in half-duplex mode</td>
</tr>
<tr>
<td>HANGUP</td>
<td>Modem is hung up when terminal logs out</td>
</tr>
<tr>
<td>HOSTSYNC</td>
<td>Host system is synchronized to terminal</td>
</tr>
<tr>
<td>INSERT</td>
<td>Default mode is insert instead of overstrike</td>
</tr>
<tr>
<td>LINESIZE</td>
<td>Sets terminal line size</td>
</tr>
<tr>
<td>LOCALECHO</td>
<td>Command line read operations are echoed</td>
</tr>
<tr>
<td>LOWER</td>
<td>Terminal accepts lowercase characters</td>
</tr>
<tr>
<td>Characteristic</td>
<td>Explanation</td>
</tr>
<tr>
<td>----------------</td>
<td>--------------------------------------------------</td>
</tr>
<tr>
<td>MAILBOX</td>
<td>Terminal does not use associated mailbox</td>
</tr>
<tr>
<td>MODEM</td>
<td>Terminal is connect via a modem</td>
</tr>
<tr>
<td>MODHANGUP</td>
<td>Modify hangup behavior</td>
</tr>
<tr>
<td>PAGESIZE</td>
<td>Sets terminal page size</td>
</tr>
<tr>
<td>PASSTHROUGH</td>
<td>Pass all characters unmodified/examined</td>
</tr>
<tr>
<td>PRINTER</td>
<td>Device has a printer port</td>
</tr>
<tr>
<td>READSYNC</td>
<td>Read synchronization is enabled</td>
</tr>
<tr>
<td>REGIS</td>
<td>Device supports graphics</td>
</tr>
<tr>
<td>REMOTE</td>
<td>Terminal is on a dialup line</td>
</tr>
<tr>
<td>SCOPE</td>
<td>Terminal is a video display device</td>
</tr>
<tr>
<td>SECURE</td>
<td>Device is on secure communication line</td>
</tr>
<tr>
<td>SIXEL</td>
<td>Device supports graphics</td>
</tr>
<tr>
<td>SYSPASSWORD</td>
<td>System password required at login</td>
</tr>
<tr>
<td>TAB</td>
<td>Terminal has mechanical tab</td>
</tr>
<tr>
<td>TTSYNC</td>
<td>Terminal is synchronized to host system</td>
</tr>
<tr>
<td>TYPEAHEAD</td>
<td>Terminal accepts unsolicited input</td>
</tr>
<tr>
<td>WRAPCR/LF</td>
<td>inserted for line wrap</td>
</tr>
<tr>
<td>XON</td>
<td>XON/XOFF handshaking used</td>
</tr>
</tbody>
</table>

**Example**

In the following example, the purpose of the DATA step is to turn off broadcast messages, and to force the terminal line width to be 80 characters. The old settings for these values are stored in macro variables so that they can be reset easily at a later time:

```plaintext
data _null_
  old_bc=setterm("broadcast",0);
  old_ls=setterm("linesize",80);
  call symput("saved_bc",put(old_bc,best.));
  call symput("saved_ls",put(old_ls,best.));
run;
```

**See Also**

- Function: “GETTERM” on page 299

---

**SYSGET**

*Returns the value of a specified operating-environment variable or symbol*

*Language element:* function

*Category:* special
OpenVMS specifics: operating-environment-variable is the name of a DCL symbol

Syntax

SYSGET("operating-environment-variable")

"operating-environment-variable"

specifies the name of a DCL symbol under OpenVMS. The value for operating-environment-variable must be enclosed in double quotation marks.

Details

The specified DCL symbol must be defined in OpenVMS before it is referenced in the SYSGET function. You can specify the symbol in a number of ways, such as in a DCL .COM file or at the DCL prompt before you invoke a SAS session. You cannot define a symbol either by using the SAS X command while you are in a SAS session or by using a logical name in OpenVMS.

If the value of the symbol is truncated, or if the symbol is not defined under OpenVMS, then SYSGET displays a warning message in the SAS log.

Example

This example defines two symbols in the OpenVMS environment:

```
$ PATH="QC:[GOMEZ.TESTING]"
$ USER="[GOMEZ.MYTESTS]"

data _null_
  length result2 result3 $ 40;
  SYMBOL2="PATH";
  SYMBOL3="USER";
  result2=sysget(trim(symbol2));
  result3=sysget(trim(symbol3));
  put result2= result3=;
run;
```

and then returns their values:

```
RESULT2=QC:[GOMEZ.TESTING]
RESULT3=[GOMEZ.MYTESTS]
```

See Also

- SYSGET function in SAS Language Reference: Dictionary
- Function: “GETSYM” on page 298
- Command: “X” on page 235

TERMIN

Allows simple input from SYS$INPUT
Functions and CALL Routines

TERMOUT

Language element: function
Category: terminal-access
OpenVMS specifics: All aspects are host-specific

Syntax

TERMOUT(prompt)

prompt

is the prompt printed on the display. It can be a character variable, a character literal enclosed in single or double quotation marks, or another character expression.

Details

The TERMIN function is easier to use than the TTOPEN, TTREAD, and TTCLOSE functions, but it does not offer the same flexibility. The return value is the characters that the user entered at the terminal. The TERMIN function accepts a maximum of 132 characters.

See Also

- Function: “TERMOUT” on page 311
- Function: “TTCLOSE” on page 312
- Function: “TTOPEN” on page 314
- Function: “TTREAD” on page 316

TERMOUT

Allows simple output to SYS$OUTPUT

Language element: function
Category: terminal-access
OpenVMS specifics: All aspects are host-specific

Syntax

TERMOUT(output)

output

is a character string to write to SYS$OUTPUT. It can be a character variable, character literal enclosed in single or double quotation marks, or another character expression.
Details
The TERMOUT function is easier to use than the TTOPEN, TTWRITE, and TTCLOSE functions, but it does not offer the same flexibility. If the TERMOUT function executes successfully, the return value is 0. Otherwise, the return value is the OpenVMS error code that indicates why it failed.

See Also

- Function: “TERMIN” on page 310
- Function: “TTCLOSE” on page 312
- Function: “TTOPEN” on page 314
- Function: “TTWRITE” on page 317

TRANSLATE

Replaces specific characters in a character expression
Language element: function
Category: character
OpenVMS specifics: Pairs of to and from arguments are optional

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

source
specifies the SAS expression that contains the original character value.

to
specifies the characters that you want TRANSLATE to use as substitutes.

from
specifies the characters that you want TRANSLATE to replace.

Details
Under OpenVMS, you do not need to provide pairs of to and from arguments. However, if you do not use pairs, you must supply a comma as a place holder.

See Also

- TRANSLATE function in SAS Language Reference: Dictionary

TTCLOSE

Closes a channel that was previously assigned by TTOPEN
Functions and CALL Routines

**TTCLOSE**

Syntax

`TTCLOSE(channel)`

**channel**

is the channel variable returned from the TTOPEN function.

**Details**

If the TTCLOSE function executes successfully, the return value is 0. Otherwise, the return value is the OpenVMS error code that indicates why it failed.

**See Also**

- Function: “TTOPEN” on page 314

**TTCONTRL**

Modifies the characteristics of a channel that was previously assigned by TTOPEN

Syntax

`TTCONTRL(control-specification,channel)`

**control-specification**

is the control string as described for the TTOPEN function. The syntax for control-specification is the same as for TTOPEN, except that the DEVICE= attribute cannot be changed. The new characteristics take effect on the next I/O operation.

**channel**

is the channel variable that was returned from the TTOPEN function.

**Details**

If the TTCONTRL function executes successfully, the return value is 0. Otherwise, the return value is the OpenVMS error code that indicates why it failed.
Example

The following example prompts the user for the password, reads the password (without echoing it to the terminal), and then writes out the password. The last step closes the channel:

```plaintext
length string $ 80;
input=' ';
chan=0;
rc=ttopen("device=tt echo",chan);
rc=ttwrite(chan,"0D0A"X||"Enter password: ");
rc=ttctrl("noecho",chan);
rc=ttread(chan,input);
rc=ttctrl("echo",chan);
rc=ttwrite(chan,"0D0A"X11"Password was: "||input);
rc=ttclose(chan);
```

See Also

- Function: “TTOPEN” on page 314

TTOPEN

Assigns an I/O channel to a terminal

Language element: function

Category: terminal-access

OpenVMS specifics: All aspects are host-specific

Syntax

`TTOPEN(control-specification,channel)`

control-specification

is the control string that specifies the terminal and processing options, separated from each other by blanks. It can be a character variable, a character literal enclosed in double quotation marks, or another character expression. The value for control-specification gives the device name and processing options and has the following form:

```
DEVICE= name <processing-option-list>
```

Each argument can be abbreviated to the shortest unique spelling. There is no default.

name

specifies the terminal name for subsequent I/O operations. DEVICE=name is required.

processing-option-list

can be one or more of the following, separated by blanks:
If you specify BUFFERFULL as one of the processing options when you enumerate the control string for the TTOPEN function, input terminates when the buffer is full or when a terminating character (either the default character or the character set with the TERMINATOR processing option) is read.

The following list enumerates the effects on input termination when you turn on combinations of processing options:

**BUFFERFULL** and **TERMINATOR**=

- causes input to be terminated when either of the following is true:
  - the buffer is full
  - the terminator string is encountered.

**NOBUFFERFULL** and **TERMINATOR**=

- causes input to be terminated only when the terminator string is encountered.

**BUFFERFULL** (only)

- causes input to be terminated when either of the following is true:
  - the buffer is full
  - you press RETURN.

**NOBUFFERFULL** (only)

- causes input to be terminated only when you press RETURN.

**TERMINATOR**=(only)

- causes input to be terminated only when the terminator string is encountered.

The default is **NOBUFFERFULL**.

**ECHO | NOECHO**

indicates whether data typed at the terminal are echoed on the display. If this attribute is not set, the behavior is based on the LOCALECHO characteristic for the terminal specified with **DEVICE**=.

The following DCL command shows the characteristics for the terminal:

```
$ SHOW TERMINAL name
```

**SIZE =**n

sets the size of the input buffer (that is, the number of characters that can be read at one time). The value can be no more than 200, the maximum size of a character variable in the SAS System. The default is 200 characters.

**TERMINATOR**=hex-string

specifies the list of characters that are considered to be terminating characters for input. Hex-string consists of hexadecimal digit pairs that correspond to the ASCII value of the characters used as terminators. Do not separate the digit pairs with delimiters such as commas or spaces.

The terminator character is used only if **NOBUFFERFULL** is set. If **NOBUFFERFULL** is in effect, the default terminator is a carriage return (hexadecimal value is 0D). If **BUFFERFULL** is specified, there is no terminator character and the input is terminated only when the buffer is full.

**TIMEOUT =**n

specifies how many seconds to wait for input from the terminal. If no input is received in the time specified, the operation fails with a time-out error. By default, there is no time limit and the input operation waits forever.
**Example**

The following example reads up to 20 characters, discarding extra characters when the buffer is full and accepting either the carriage return or the horizontal tab character (hexadecimal value is 09) as terminators. If the read is successful, the program prints the string:

```plaintext
length string $ 20;
rc=ttopen("dev=TT: size=20 term=0D09",chan);
rc=ttread(chan,string,size);
if size>0 & rc=0 then put string;
rc=ttclose(chan);
```

**See Also**

- Function: “TTCLOSE” on page 312

---

**TTREAD**

**Reads characters from the channel assigned by TTOPEN**

**Language element:** function

**Category:** terminal-access

**OpenVMS specifics:** All aspects are host-specific

**Syntax**

```
TTREAD(channel,buffer,<size>)
```

**channel**

is the channel variable returned from the TTOPEN function.

**buffer**

is the character variable where the returned characters are stored.

**size**

is an optional numeric parameter which indicates the maximum number of characters to read and receives the number of characters read. If you do not specify
size, the TTREAD function reads characters up to the size of buffer. The handling of extra characters is determined by the BUFFERFULL option specified with the TTOPEN function.

Details
If the TTREAD function executes successfully, the return value is 0. Otherwise, the return value is the OpenVMS error code that indicates why it failed.

See Also

- Function: “TTOPEN” on page 314

---

**TTWRITE**

**Writes characters to the channel assigned by TTOPEN**

Language element: function  
Category: terminal-access  
OpenVMS specifics: All aspects are host-specific

**Syntax**

```
TTWRITE(channel, buffer, <size>)
```

**channel**

is the channel variable returned from the TTOPEN function.

**buffer**

is the character string variable that contains the data to be written.

**size**

is an optional numeric parameter that specifies how many characters to write from buffer. If you do not specify size, the entire buffer is sent, including any trailing blanks.

Details
If the TTWRITE function executes successfully, the return value is 0. Otherwise, the return value is the OpenVMS error code that indicates why it failed.

The TTWRITE function does not supply any carriage control. You must insert into the buffer any carriage-control characters that you want.
See Also

- Function: “TTOPEN” on page 314

---

**VMS**

Spawns a subprocess and executes a DCL command

**Language element:** function

**Category:** general-purpose OpenVMS

**OpenVMS specifics:** All aspects are host-specific

**Syntax**

\[
RC = VMS(DCL-command)
\]

**RC**

is the return code. If the VMS function executes successfully, the return value is 0. Otherwise, the return value is the OpenVMS error code that indicates why it failed. If you supply an invalid command, you will receive a return error code like the following:

229520 %CLI-W-IVVERB, unrecognized command verb - check validity and spelling.

**DCL-command**

is the DCL command that is passed to the subprocess. It can be a character variable, a character literal enclosed in quotation marks, or another character expression.

**Details**

The VMS function spawns a subprocess and executes the command that is passed to it. Any output that is produced is sent to SYS$OUTPUT. If you are using the SAS windowing environment, the results appear in a new window. To close the new window, select the **File** menu and then select **Exit**. This is consistent with the behavior of the X statement and the X command.

If the VMS function executes successfully, the return value is 0. Otherwise, the return value is an OpenVMS error code that indicates why the function failed.

**Comparisons**

The VMS function is similar to the X statement, the X command, the %SYSEXEC macro, and the CALL SYSTEM routine. In most cases, the X statement, the X command, or the %SYSEXEC macro are preferable because they require less overhead. However, the VMS function is useful for conditional processing because it returns a return code. The CALL SYSTEM routine can be useful in certain situations because it is executable, and because it accepts expressions as arguments.
See Also

- "Issuing DCL Commands during a SAS Session" on page 37
- Statement: “X” on page 384
- Command: “X” on page 235
- CALL routine: “CALL SYSTEM” on page 276
- %SYSEXEC macro in "Macro Statements" on page 463