

# An Autoexec Companion, Allocating Location Names during Startup

Ronald J. Fehd, Stakana Analytics

## Abstract

**Description :** Like other computer languages SAS® software provides a method to automatically execute statements during startup of a program or session. This paper examines the names of locations chosen in the filename and libname statements and the placement of those names in options that enable all programs in a project to have standardized access to format and macro catalogs, data sets of function definitions and folders containing reusable programs and macros. It also shows the use of the global symbol table to provide variables for document design.

**Purpose :** The purpose of this paper is to examine the default values of options, suggest naming conventions where missing, and provide both an example autoexec and a program to test it.

**Audience :** programmers, all levels

**Keywords :** document design (title, footnote),  
environment and macro variables,  
file searches: library (*librefs*), programs (*filerefs*),  
format search (fmtsearch),  
functions (cexist, exist, ifc, getoption, %sysfunc),  
macro autocall: file search (mautosource, sasautos),  
macros, catalog of compiled and stored (mstored, sasmstore),  
procedure fcmp data set of functions (cmplib)

## In this paper

<b>Introduction</b>	<b>2</b>
<b>Allocating variables in the global symbol table during startup</b>	<b>5</b>
Global variables for output . . . . .	6
Filerefs for input, programs and autocall macros . . . . .	7
Catrefs and librefs for format and macro catalogs, data sets and functions . . . . .	8
Other options . . . . .	10
<b>Programs</b>	<b>11</b>
Autoexec.sas . . . . .	11
Autoexec-test.* . . . . .	12
<b>Summary</b>	<b>15</b>
<b>Bibliography</b>	<b>16</b>

---

## Introduction

### Overview

SAS software provides a facility to automatically execute statements during startup of a program. The option name is `autoexec`. Its default value is `autoexec.sas`, in the Windows operating system. If a file with this name is present in the startup folder then it is `%included` before the program.

The `autoexec` file provides all programs in the project folder the same list of asset names and locations. Names of locations are allocated with the `filename` and `libname` statements. These location names, referred to as *filerefs* and *librefs* respectively, are arbitrary. Whichever names are chosen, the next step is to insert them into the corresponding option which enables SAS software to search for a particular object within the asset.

### the matrix

#### Options and Location Names

<u>asset</u>	<u>option</u>	<u>value</u>	<u>default</u>	<u>object</u>	<u>in</u>	<u>see also</u>
functions	<code>cmplib</code>	<i>libref.data</i>	n/a	definitions	data set	
formats	<code>fmtsearch</code>	<i>librefs</i>	work library	formats	catalog	<code>fmterr</code>
macros						
autocall	<code>sasautos</code>	<i>filerefs</i>	<code>sasautos</code>	files	folder	<code>mautosource</code>
compiled	<code>sasmstore</code>	<i>libref</i>	n/a	definitions	catalog	<code>mstored</code>

---

The introduction has these topics.

- style guide
- naming conventions
- definitions
- folder structure
- syntax for lists

---

### style guide

This paper uses *italics* and `text` fonts to identify specific uses of words.

*italics* : SAS documentation words, e.g. *fileref*, *libref*

`text` : SAS keyword, e.g. `filename`, `libname`

---

### naming conventions

This is the list of naming conventions used in this papers

site : folders available to all projects are referred to with a prefix of 'site\_'

project : the 'logical name' for the folder containing \*.sas programs is 'project'

I/O : other folders have a prefix according to function:

input: 'in\_', output: 'out\_',

---

## definitions

This is a list of the terms used here.

addressing :	a method of referring to another folder
absolute :	provide a full (direct) path to the folder using either a drive letter or universal-naming convention (UNC), e.g., 'c:\temp' also known as 'hard-coded'
relative :	provide an indirect reference to a sibling or parent folder e.g., '..\csv' also known as 'soft-coded'
	see paragraph 'folder structure' below
directory :	see folder; in SAS documentation, see 'directory-specification'
<i>fileref</i> :	is a 'logical name' for one or more folders; first argument of the <code>filename</code> statement; the value of a <i>fileref</i> is also called a 'directory-specification'
folder :	a folder contains files and may contain sub-folders (or child-folders); as used in this paper folders contain text files; also called a directory
library :	is one or more folders; in SAS documentation 'library' generally refers to folder(s) containing files maintained by SAS software, i.e., catalogs and data sets
<i>libref</i> :	is a 'logical name' for a library, first argument of the <code>libname</code> statement, the value of a <i>libref</i> is also called a 'directory-specification'
'logical name' :	is a phrase used in SAS documentation; here it means a variable name in the global symbol table

## folder structure

The assignments discussed in this paper are based on these assumptions; your corporate culture may differ.

The purpose of this structure is to support copying of programs from a working project to a new project and minimize changes necessary to get the programs working.

## folder naming conventions

...\project-A\	csv	comma separated values
	dat	data files
	out-pdf	vendor-specific files
	out-rtf	vendor-specific files
	sas	*.sas programs
	sas7b	data sets, catalogs
...\project-B\	like project-A	
...\SAS-site\		
	includes	*.sas programs
	macros	*.sas programs
	sas7b	data sets, catalogs
	sas-include-tests	*.sas programs

## syntax for lists

A list contains one or more items. In the `catname`, `filename`, `libname` and `options` statements a value may be a single item or a list of items which must be enclosed in parentheses, e.g.,

item: <code>filename project '..'</code> ;	one folder
list: <code>options sasautos</code> <code>= (project site_mac sasautos);</code>	three <i>filerefs</i>

---

## Programs for discovery during research

---

### Overview

These programs were used in research for this paper.

- options define value
  - options groups
  - sql dictionary options
  - finding word in options
  - getoption: fetching value
- 

### options define value

The define and value options are used to display information about an option in the options procedure.

```
PROC options define value option = autoexec;
```

---

### define groups

This program shows the groups of the options discussed here.

```
PROC options group = envfiles;*autoexec fmtsearch sasautos;  
PROC options group = macro    ;*mautosource mstored sasmstore;  
PROC options group = sasfiles;*cmplib;
```

---

### sql dictionary options

The sql procedure offers another avenue of discovery.

```
PROC sql; describe table dictionary.options;  
quit;  
  
PROC sql; select distinct group  
            from dictionary.options;  
quit;
```

---

### finding word in options

The sql procedure can be used to search for a word in either the option name or the option description.

```
*name: find-word-in-options.sas;  
%let word = macro;  
PROC sql; select group, optname, optdesc  
            from dictionary.options  
            where index(upcase(optname),"%upcase(&word)")  
                  or index(upcase(optdesc),"%upcase(&word)");  
quit;
```

---

## getoption: fetching value

The `getoption` function can be called in open code by the `%sysfunc` function. There are three types of options: boolean, character and numeric. Boolean options use the prefix 'no' before the option name for false. Character and numeric option values can be fetched with the keyword `option`. This returns the value in the form: `option-name=value`. This value can be used in a macro variable assignment statement.

---

## sysfunc getoption

```
*name: sysfunc-getoption.sas;
**** option type character;
%let %sysfunc(getoption(autoexec,keyword));
%put echo &=autoexec;
**** option type boolean in (T=<opt-name>,F=no<opt-name>);
%put mautosource=%sysfunc(getoption(mautosource));
options nomautosource;
%put mautosource=%sysfunc(getoption(mautosource));

log: echo AUTOEXEC=C:\...\autoexec-companion\sas\autoexec.sas
5 %put mautosource=%sysfunc(getoption(mautosource));
mautosource=MAUTOSOURCE
6 options nomautosource;
7 %put mautosource=%sysfunc(getoption(mautosource));
mautosource=NOMAUTOSOURCE
```

---

---

## Allocating variables in the global symbol table during startup

---

### Overview

This section describes the allocations of variable names and values in the global symbol table that are used in programs.

As used in this paper, the term 'global symbol table' has these entities.

- environment variables
  - *catrefs* concatenate catalog names
  - *filerefs* location names
  - *librefs* location names
  - macro definitions `%let mvar=...;`
  - macro variables `%macro ... %mend;`
  - options
  - running text: titles and footnotes
-

---

## Global variables for output

---

### Overview

Managing output has two aspects, marking up the document and where to write the document, i.e., the destination folder.

- document design
  - output destination
- 

### document design

Output documents may have running text, which consists of either or both of headers or footers. The global symbol table contains a maximum of ten entries for `title` or `footnote` assignments.

naming conventions →

The naming conventions used here are `title1` contains the project name and `title2` contain the name of the program; titles 3–10 are assigned and cleared by programs.

```
title1 'An autoexec.sas Companion';
*** sysin contains path\program-name.sas;
title2 "%sysfunc(getoption(sysin))";
           * show program-name only;
title2 "%scan(%sysfunc(getoption(sysin)),-1,/\\)";
```

---

### output destination

All output may be directed to a single folder with the use of an environment variable. A bang or exclamation point (!) is used to refer to an environment variable.

```
***** allocate environment variables;
*** relative: sibling folder;
options set = out_pdf '..\out-pdf';
```

usage: `ods pdf file = '!out_pdf\report.pdf';`

naming conventions →

Refer to the paragraph 'folder structure' above, on page 3.

commentary

Other authors use macro variables for this task.

---

---

## Filerefs for input, programs and autocall macros

---

### Overview

One important use of the global symbol table is for names of locations of reusable programs and input text. This section provides suggestions for naming of the following items.

- filerefs for input
- filerefs for programs
- autocall macros

---

### filerefs for input

Use 'in\_' as the prefix of names for folders containing text files of input. These example shows the folders are sibling folders and are described with relative addressing.

```
filename in_csv '..\csv';  
filename in_dat '..\dat';
```

naming conventions → Refer to the paragraph 'folder structure' above, on page 3.

---

### filerefs for programs

Project programs may reuse other programs in the project folder as well as utility programs in the site folder.

naming conventions → Use 'site\_' as the prefix of location names in the site folder.

```
filename project '.';                * here: startup folder;  
  
    *** allocate environment variables (e-var);  
                                * absolute: hard-coded;  
options set = site_root 'C:\SAS-projects\SAS-site';  
    *** use bang (!) as prefix to reference an e-var;  
filename site_inc "!site_root\includes"; * double quotes;  
filename site_mac '!site_root\macros\'; *or single quotes;
```

- notes:**
- This example shows the environment variable 'site\_root' is hard-coded with a path to a folder that is not a sibling of the project folder.
  - Either single or double quotes may be used.

---

### autocall macros

There are a pair of options for *autocall* macros; *mautosource* is boolean, *sasautos* is character and is a list of *filerefs*. This options statment sets up the *autocall* macro search facility for three *filerefs*: project, site\_mac, and the SAS-supplied *fileref* sasautos.

```
** pair macro autocall: boolean, search-path filerefs;  
options mautosource  
    sasautos = (project site_mac sasautos);
```

**note:** The list of filerefs must be enclosed in parentheses.

---

---

## Catrefs and librefs for format and macro catalogs, data sets and functions

---

### Overview

This section describes naming convention choices and statements to allocate location names for SAS-managed files such as catalogs of formats, compiled macros, and data sets containing functions.

- choose libref library
- catalogs for formats
- catalogs for macros
- catrefs to concatenate catalogs
- data sets for fcmp functions

---

### choose libref library

There are two reasons to use the *libref* library.

- SAS uses the term throughout its documentation
- the default search path for format catalogs is library

---

### catalogs for formats

The `format` procedure compiles `value` statements and other definitions and saves into the the *libref* named in the `library` option of the procedure statement.

syntax: `libname libref 'folder';`

usage: `libname library '..\sas7b'; * sibling folder;`

**note:** The statement `proc format library = library;` creates a catalog in the *libref* library.

naming conventions →

The catalog name is `library.formats`.

*Libref* library is the second *libref* in the default format-search path.

Examine the format search path with this statement.

```
%put %sysfunc(getoption(fmtsearch,keyword));
```

```
log: fmtsearch=(WORK LIBRARY)
```

beware naming collisions

Do not change the format search path to `(library work)` because:

- it is frustrating for users to create formats, which are stored in the *libref* `work`, that are superseded by formats in the *libref* library
- it is extremely difficult to debug this problem

**note:** The format catalogs in *librefs* `work` and `library` are always searched even if they are not in the `fmtsearch` list.

---



## catalogs for compiled macros

Macros may be compiled and stored in a catalog.  
Use the macro statement option `store` to save in a catalog.

example: `%macro demo(...)/store;`

There are a pair of options for compiled and stored macros;  
`mstored` is boolean, `sasmstore` is character and is one *libref*.

```
libname site_lib '!site_root\sas7b';  
** pair macros compiled and stored: boolean, libref;  
options mstored sasmstore = site_lib;* or library;
```

**note:** The name of the catalog where compiled and stored macro definitions are saved is *libref.sasmacr*. The search sequence is *libref* work where autocall macros are saved, then *libref* of option `sasmstore`; this search sequence cannot be changed. Compare to the option `fmtsearch` on page 8.

naming conventions → Be aware of naming collisions — masking — when using compiled and stored macros, i.e., macros with the same name in both the *work* and *sasmstore* catalogs.

SASautos Companion [6] has a utility program, `listmcat`, to find naming collisions in macro catalogs.

---

## catrefs of multiple catalogs

The `catname` statement can be used to concatenate catalogs of either formats or macros.

```
*setup: allocate librefs;  
libname lib_xtra '..\sas7b';  
libname lib_site '..\..\SAS-site\sas7b';  
*usage: allocate catref for two or more catalogs;  
catname cat_fmtns (lib_xtra.formats lib_site.formats);  
options fmtsearch = (work library cat_fmtns);
```

---

## data set for fcmp functions

There are a pair of statements for functions created by the `fcmp` procedure.  
`libname` allocates a *libref*; option `cmplib` is character;  
it is a two-level data set name, e.g., *libref.data*.

naming conventions → Both the *libref* and data set name are arbitrary.

```
*creation: proc fcmp outlib=libref.data_set_name.package;  
***** functions compiled by proc fcmp;  
options cmplib = site_lib.functions; *or library.functions;
```

---

## Other options

---

### Overview

There are several hundred options many of which may be set in the autoexec. See the paragraph Suggested Reading below for other authors' recommendations.

```
options fullstimer      /* max info in timer block;
      msglevel = i      /* more messages in log   ;
      nofmterr;         * no err: formats missing;
```

### append, insert

Many of the above options can be modified with either or both of the options append or insert.

```
*syntax: options insert|append=(opt-name=libref|'..\path');
*example;
options insert=(fmtsearch=lib_a);*libref;
*discovery;
proc options option=fmtsearch value;
```

**note:** In SAS documentation see the pages *APPEND= System Option*, *INSERT= System Option*, and *Using SAS System Options*, paragraphs: *Option Value Information for an Option Modified by the INSERT and APPEND Options* and *Changing an Option Value by Using the INSERT and APPEND System Options*.

---

---

## Programs

### Overview

This section contains the listings of an `autoexec.sas` and test programs.

`autoexec.sas`

---

```
title1  'An autoexec.sas Companion';
        %let %sysfunc(getoption(sysin,keyword));
        %put echo &=sysin;
title2  "&sysin";          *** path\program-name.sas;
title2  "%scan(&sysin,-2,/\..)"; *** program-name only;

        *** allocate environment variables;
        *** relative: sibling folder;
options set = out_pdf  "..\out-pdf";    * double quotes;
options set = out_rtf  "..\out-rtf";    *or single quotes;
        *** absolute: hard-coded compare to site_lib;
options set = site_root 'C:\SAS-projects\SAS-site';

        *** various folders with input files;
filename in_csv  '..\csv';
filename in_dat  '..\dat';
filename project '.'; * here: startup folder: *.sas programs;
        *** use bang (!) as prefix to reference an e-var;
filename site_inc "!site_root\includes"; * double quotes;
filename site_mac '!site_root\macros' ; *or single quotes;

libname library  '..\sas7b';            *sibling folder;
*libname site_lib '!site_root\sas7b';
libname site_lib '..\..\SAS-site\sas7b';*compare to site_root;

*catname catref ...;

        *** functions compiled by proc fcmp in data set;
options cmplib = library.functions;*or site_lib.functions;

        *** format catalogs, see also option fmterr;
options fmtsearch = (work library site_lib.formats);

** pair macro autocall: boolean, search-path filerefs;
options mautosource
        sasautos = (project site_mac sasautos);

** pair macros compiled and stored: boolean, one libref;
options mstored sasmstore = library;    *or site_lib;

options nofmterr          %* no err msg: fmt missing;
```

---

## Overview

<code>cexist :</code>	exist catalog
<code>exist :</code>	exist libref.data-set
<code>getoption :</code>	returns value of an option
<code>ifc :</code>	evaluate condition, return character string
<code>%nrstr :</code>	no rescan string, delays evaluation of references in argument
<code>%sysfunc :</code>	system function, allows access to certain data step functions; used here with <code>ifc</code> to conditionally execute statements
<code>%sysfunc(ifc :</code>	<code>%sysfunc(ifc(condition</code>
	<code>,%nrstr(%put true ;)</code>
	<code>,%nrstr(%put false;))</code>
	evaluate a condition in open code, return either true statement(s) or false statement(s); the <code>%nrstr</code> function guarantees only one statement is returned
<code>%let catalog= :</code>	macro variable assignment statement
<code>exist_catalog :</code>	<code>%let exist_catalog = %nrstr(echo</code>
	<code>%sysfunc(ifc(%sysfunc(cexist(&amp;catalog))</code>
	<code>,,%nrstr(not ),))exist &amp;catalog);</code>
	this macro variable contains a function call to verify the existence of the catalog named in the macro variable <code>catalog</code>
<code>%unquote :</code>	<code>%let catalog = work.formats;</code>
	<code>%put %unquote(&amp;exist_catalog);</code>
	resolves the macro variable references and function calls in the macro variable <code>exist_catalog</code>

This batch file calls SAS software.

The DOS command 'call' allows multiple instances in another batch file. The string percent+asterisk '%\*' passes all command-line parameters on to sas.exe.  
See usage in autoexec-test.bat below.

```
/* name: <project-name>\sas\sasv9.cfg */
-sasinitialfolder '.' /* set 'file-open' to startup folder */
```

12

## autoexec-test.bat

This is the batch file used to test an autoexec. The startup-only option echoauto is used to echo the statements in the autoexec; its action is similar to `%include 'autoexec.sas'\source2;`

```
rem name: autoexec-test.bat
sas      autoexec-test -echoauto
```

## autoexec-test.sas

```
**** name: autoexec-test.sas;
**** many options are set in configuration files;
%put echo sascfg=%sysget(sascfg);
%put echo %sysfunc(getoption(config,keyword));

*command-line option sasinitialfolder may be set by icons;
*or project config, compare to folder of fileref project;
%put echo %sysfunc(getoption(sasinitialfolder,keyword));

*review all filerefs;
filename _all_ list;

**** review cmplib=libref.data-set-name of functions;
%let %sysfunc(getoption(cmplib,keyword));
%sysfunc(ifc(%sysfunc(exist(&cmplib))
            ,%nrstr(%put echo exist(&cmplib);)
            ,%nrstr(%put echo not exist(&cmplib);),))

%let exist_catalog = %nrstr(echo
%sysfunc(ifc(%sysfunc(cexist(&catalog))
            ,%nrstr(not ),)exist &catalog);

**** formats: search list of librefs;
%let %sysfunc(getoption(fmtsearch,keyword));*remove parens;
%let fmtsearch=%substr(&fmtsearch,2,%length(&fmtsearch)-2);
%put echo &=fmtsearch;
**** check default = work library;
%let catalog = work.formats;
%put %unquote(&exist_catalog);
%let catalog = library.formats;
%put %unquote(&exist_catalog);
**** remove default librefs WORK and LIBRARY;
%let fmtsearch=%sysfunc(tranwrd(&fmtsearch,WORK,%str( )));
%let fmtsearch=%sysfunc(tranwrd(&fmtsearch,LIBRARY,%str( )));
**** there may be one, or more;
%let fmt_lib = %scan(&fmtsearch,1,%str( ));
**** if fmt-lib has dot then is two-level name;
%let catalog = %sysfunc(ifc(%index(&fmt_lib,.)
            ,%nrstr(&fmt_lib),%nrstr(&fmt_lib..formats) ));
%put %unquote(&exist_catalog);

**** macro autocall;
%put echo mautosource=%sysfunc(getoption(mautosource));
%put echo %sysfunc(getoption(sasautos,keyword));
*allocate fileref sasautos to check if modified in config;
%include sasautos(af)/nosource2;
%put echo sasroot=%sysget(sasroot);
filename sasautos list;
**** if sasautos contains paths other than sasroot
    then review configuration file(s) listed above;

**** macros compiled and stored;
%let mstored =%sysfunc(getoption(mstored));
%let sasmstore=%sysfunc(ifc(&mstored eq MSTORED
            ,%nrstr(%sysfunc(getoption(sasmstore))),%nrstr(work)));
%put echo &=mstored &=sasmstore;
%let catalog = &sasmstore..sasmacr;
%put %unquote(&exist_catalog);
```

## autoexec-test-log.txt

---

```
NOTE: AUTOEXEC processing beginning; file is
C:/.../autoexec-companion/sas/autoexec.sas.
19 libname library '..\sas7b';
NOTE: Libref LIBRARY was successfully assigned as follows:
Physical Name= C:\...\autoexec-companion\sas7b
NOTE: AUTOEXEC processing completed.
1 **** name: autoexec-test.sas;
2 **** many options are set in configuration files;
3 %put echo sascfg=%sysget(sascfg);
echo: sascfg=C:\...\SASHome\SASFoundation\9.4\nls\en
4 %put echo %sysfunc(getoption(config,keyword));
echo: CONFIG=(C:\...\SASHome\SASFoundation\9.4\sasv9.cfg"
"C:\...\SASHome\SASFoundation\9.4\nls\en\sasv9.cfg"
"C:\...\autoexec-companion\sas\sasv9.cfg" )
8 %put echo %sysfunc(getoption(sasinitialfolder,keyword));
echo: SASINITIALFOLDER=
11 filename _all_ list;
...
NOTE: Fileref= PROJECT
Physical Name= C:\...\autoexec-companion\sas
NOTE: Fileref= SITE_INC
Physical Name= C:\SAS-projects\SAS-site\includes
NOTE: Fileref= SITE_MAC
Physical Name= C:\SAS-projects\SAS-site\macros
13 **** review cmlib=libref.data-set-name of functions;
14 %let %sysfunc(getoption(cmlib,keyword));
15 %sysfunc(ifc(%sysfunc(exist(&cmlib))
16 ,%nrstr(%put echo exist(&cmlib);)
17 ,%nrstr(%put echo not exist(&cmlib);),))
echo: not exist(library.functions)
19 %let exist_catalog = %nrstr(echo
20 %sysfunc(ifc(%sysfunc(cexist(&catalog))
21 ,,%nrstr(not ),))exist &catalog);
23 **** formats: search list of librefs;
24 %let %sysfunc(getoption(fmtsearch,keyword));*remove parens;
25 %let fmtsearch=%substr(&fmtsearch,2,%length(&fmtsearch)-2);
26 %put echo &=fmtsearch;
echo: FMTSEARCH=WORK LIBRARY SITE_LIB.FORMATS
27 **** check default = work library;
28 %let catalog = work.formats;
29 %put %unquote(&exist_catalog);
echo: not exist work.formats
30 %let catalog = library.formats;
31 %put %unquote(&exist_catalog);
echo: not exist library.formats
32 **** remove default librefs WORK and LIBRARY;
33 %let fmtsearch=%sysfunc(tranwrd(&fmtsearch,WORK,%str( )));
34 %let fmtsearch=%sysfunc(tranwrd(&fmtsearch,LIBRARY,%str( )));
35 **** there may be one, or more;
36 %let fmt_lib = %scan(&fmtsearch,1,%str( ));
37 **** if fmt-lib has dot then is two-level name;
38 %let catalog = %sysfunc(ifc(%index(&fmt_lib,.)
39 ,%nrstr(&fmt_lib),%nrstr(&fmt_lib..formats) ));
40 %put %unquote(&exist_catalog);
echo: not exist SITE_LIB.FORMATS
42 **** macro autocall;
43 %put echo mautosource=%sysfunc(getoption(mautosource));
echo: mautosource=MAUTOSOURCE
44 %put echo %sysfunc(getoption(sasautos,keyword));
echo: SASAUTOS=(project site_mac sasautos)
...
72 %put echo sasroot=%sysget(sasroot);
echo: sasroot=C:\...\SASHome\SASFoundation\9.4
73 filename sasautos list;
NOTE: Fileref= SASAUTOS
Physical Name= C:\...\SASHome\SASFoundation\9.4\core\sasmacro
...
Physical Name= C:\...\SASHome\SASFoundation\9.4\stat\sasmacro
77 **** macros compiled and stored;
78 %let mstored =%sysfunc(getoption(mstored));
79 %let sasmstore=%sysfunc(ifc(&mstored eq MSTORED
80 ,%nrstr(%sysfunc(getoption(sasmstore))),%nrstr(work) ));
81 %put echo &=mstored &=sasmstore;
echo: MSTORED=MSTORED SASMSTORE=library
82 %let catalog = &sasmstore..sasmacr;
83 %put %unquote(&exist_catalog);
echo: not exist library.sasmacr
```

---

---

## Summary

### Conclusion

An autoexec file provides the ability to centralize statements which manage entries into and relations within the global symbol table. This centralization encourages standardization and reduces the complexity, and therefore the testing burden, of all programs in the project.

---

### Suggested Reading

- predecessor : [6] SASautos Companion, Fehd [5]
- autoexec : Cogswell [3] describes a robust set of Windows batch programs that uses `systask` with error checking and notification, shows directory structure, how to restart, provides method of debugging (testing) in individual programs.  
Fang and Gorrell [4] discuss use of macro variables and the `%include` statement;  
Carpenter and Smith [2] discuss naming conventions of directories, files and libraries; use macro variables for paths, production/testing and titles; provide paths in option `sasautos`; shows two *librefs* for compiled-and-stored macro option `sasmstore`  
Taubman [11] shows `filename` statement with `pipe` option in an autoexec, uses environment variables
- batch : Walsh [13] shows differences between program development in session vs. batch and why an autoexec is necessary for batch
- formats : Gudavalli [7]
- functions : user-written functions, `proc fcmp`, Carpenter [1], Secosky [10]
- libname : William E. Benjamin Jr [14]
- options : Heaton [8], Poll [9], Thornton and Barbalau [12]

---

### Author information

#### Ronald J. Fehd

<mailto:Ron.Fehd.macro.maven@gmail.com>

sco.wiki

[http://www.sascommunity.org/wiki/Ronald\\_J.\\_Fehd](http://www.sascommunity.org/wiki/Ronald_J._Fehd)

LinkedIn

[www.linkedin.com/Ronald.Fehd](http://www.linkedin.com/Ronald.Fehd)

affiliation

Stakana Analytics, Senior Maverick

also known as

macro maven on SAS-L, Theoretical Programmer

Programs:

[http://www.sascommunity.org/wiki/Autoexec\\_Companion](http://www.sascommunity.org/wiki/Autoexec_Companion)

previous publication:

<http://www.lexjansen.com/mwsug/2015/BB/MWSUG-2015-BB-10.pdf>

---

### Trademarks

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration. Other brand and product names are trademarks of their respective companies.

---

## Bibliography

- [1] Arthur L. Carpenter. Using proc fcmp to the fullest: Getting started and doing more. In *SAS Global Forum*, 2013. URL [support.sas.com/resources/papers/proceedings13/139-2013.pdf](http://support.sas.com/resources/papers/proceedings13/139-2013.pdf). 10 pp.; user-written functions, routines and subroutines with proc fcmp, arguments, returning multiple character or numeric values, using sysfunc to call.
  - [2] Arthur L. Carpenter and Richard O. Smith. Library and file management: Building a dynamic application. In *SUGI-27*, 2002. URL <http://www2.sas.com/proceedings/sugi27/p021-27.pdf>. 8 pp.; ApDev; autoexec, directory naming conventions, macro libraries, filerefs and librefs;.
  - [3] Denis Cogswell. More than batch — a production SAS(R) framework. In *SUGI-30*, 2005. URL <http://www2.sas.com/proceedings/sugi30/021-30.pdf>. 15 pp.; ApDev, autoexec in production environment.
  - [4] Zhengyi Fang and Paul Gorrell. Two steps to libname-free coding: Use of macro parameters and autoexec.sas. In *NESUG*, 2010. URL <http://www.lexjansen.com/nesug/nesug10/ad/ad03.pdf>. 5 pp.; using global macro variables to replace libref.data-set names.
  - [5] Editor: R.J. Fehd. ListMcat: list macro catalog. In *sasCommunity.org*, 2009. URL [http://www.sascommunity.org/wiki/ListMcat\\_List\\_Macro\\_Catalog](http://www.sascommunity.org/wiki/ListMcat_List_Macro_Catalog). program to list macro definitions in work and sasmstored catalogs.
  - [6] Ronald Fehd. A SASautos Companion: Reusing macros. In *SAS Users Group International Annual Conference Proceedings*, 2005. URL <http://www2.sas.com/proceedings/sugi30/267-30.pdf>. Tutorials, 12 pp.; topics: autocall, autoexec, configuration file (sasv9.cfg), compiled and stored macros, masking, options (mautosource, mstored, sasmstore), program reuse, sasautos (environment variable, filename, option); info: autoexec examples, utility program ListMcat: show same-named macros in different catalogs.
  - [7] Ram Gudavalli. Search across format libraries in FMTSEARCH option. In *NESUG*, 2011. URL <http://www.lexjansen.com/nesug/nesug11/pf/pf03.pdf>. 3 pp.; list or verify formats in various catalogs.
  - [8] Edward Heaton. SAS(R) system options are your friends. In *SESUG*, 2004. URL <http://analytics.ncsu.edu/sesug/2004/TU07-Heaton.pdf>. 15 pp.; nofmterr, sasinitialfolder.
  - [9] Denise Poll. Options — versatile players in the game of SAS. In *SAS Global Forum*, 2011. URL <http://support.sas.com/resources/papers/proceedings11/240-2011.pdf>. 8 pp.; Programming, Beyond Basics; system tools available to view and manage options, procedures optload and optsave, options modifiable by insert and append.
  - [10] Jason Secosky. Executing a proc from a data step. In *SAS Global Forum*, 2012. URL <http://support.sas.com/resources/papers/proceedings12/227-2012.pdf>. 12 pp.; user-written functions with proc fcmp, functions run\_macro, run\_sasfile, dosub, dosubl; proc http.
  - [11] Stephen B. Taubman. Procedures for a new SAS(R) release. In *SUGI-25*, 2000. URL <http://www2.sas.com/proceedings/sugi25/25/po/25p206.pdf>. 3 pp.; autoexec for site, filename with pipe option.
  - [12] Patrick Thornton and Iuliana Barbalau. Working the system: Our best SAS(R) options. In *SAS Global Forum*, 2012. URL <http://support.sas.com/resources/papers/proceedings12/047-2012.pdf>. 10 pp.; tools to examine options.
  - [13] Irina Walsh. Pros and cons of interactive SAS(R) mode vs. batch mode. In *WUSS*, 2010. URL [http://www.wuss.org/proceedings10/coders/2937\\_4\\_0\\_COD\\_Walsh.pdf](http://www.wuss.org/proceedings10/coders/2937_4_0_COD_Walsh.pdf). 9 pp.; uses macro variables in autoexec.sas, directory structure, sysfunc+ifc, proc printto for log and output in session, ods rtf.
  - [14] William E. Benjamin Jr. The little engine that could: Using libname engine options to enhance data transfers between SAS(R) and Microsoft Excel files. In *SESUG*, 2011. URL <http://analytics.ncsu.edu/sesug/2011/CC13.Benjamin.pdf>. 10 pp.;.
- 

Information is **the** difference  
that makes **a** difference  
Gregory Bateson, 1904–1980  
Steps to an Ecology of Mind, 1972