Paper BB-03

Which SASAUTOS Macros Are Available to My SAS® Session?

Harry Droogendyk, Stratia Consulting Inc., Lynden, ON

ABSTRACT

The SAS installation process makes a number of SAS supplied macros available to your SAS session via the SASAUTOS option. In addition, useful macros created by users in your organization may be made available to SAS users via SASAUTOS or by way of the Compiled Stored Macro facility. But… how do I know which macros are available? And, if a macro has been defined in more than one location, which one will be utilized? This paper will explain the different methods of defining and storing macros, discuss how SAS searches for macros, demonstrate how to specify the appropriate options to allow access to compiled stored macros and define the SASAUTOS search order. The meaty stuff will involve a walk through **%list_sasautos**, a useful utility macro, written by the author, that will identify all of the macros available to your SAS session.

INTRODUCTION

One of the of many efficiencies available in the SAS "toolkit" is the ability to create modular code through the use of SAS macro and make that code available to a wider group of SAS users. Typically the collection of macros will be stored in an installation or project defined folder or directory from which they'll be referenced. Using a central repository of SAS macros reduces or eliminates the problems that arise when multiple versions of the same macro exist in multiple locations. Maintenance is minimized, common functionality is easily shared and business rules are encapsulated in one place.

Three different methods are available to the SAS user seeking to take advantage of common macro stores:

1. via **%INCLUDE** of source entries
2. compiled stored macros
3. AUTOCALL libraries which can include **.SAS** modules and source catalogs.

Since the three methods can be used separately or in conjunction, another level of complexity emerges. While the MAUTOLOCDISPLAY system option does tell you from *which* AUTOCALL library the macro was compiled when invoked for the first time, there's no option or procedure to traverse the search chain looking for all macros available to your SAS session.

What happens when macros of the same name are defined in more than one location? What is the search order SAS uses when finding and invoking a SAS macro?

This paper will answer these questions, culminating in a code-walkthrough of the **%list_sasautos** utility macro.

%INCLUDE CODE

%INCLUDE can be used to incorporate *any* SAS code into *any* program. The included code might be an entire SAS program, a partial step or a macro definition.

```
%include "c:\sas\include\customer_profit_calc.sas" /source2;

    or

filename inc "c:\sas\include";
%include inc(customer_profit_calc) /source2;
```

Included code is retrieved, compiled and executed. If the module contains a macro definition (ie. code specified between **%macro** and **%mend** statements), the macro is compiled and stored in the WORK library SAS macro

catalog and made available to the SAS session. Using %INCLUDE requires that each user know the paths to the installation or project libraries.

COMPILED STORED MACROS

Before any macro is executed it must be compiled. In the %INCLUDE example above, the compiled macro is stored in the temporary WORK.SASMACR catalog. For the sake of efficiency, large and complex macros are often stored in compiled form in a SAS macro catalog also named SASMACR, but in a permanent library. Each entry in this catalog has a type of MACRO. To compile and store macros, the MSTORED option must be turned on and the permanent library in which the SASMACR catalog is to be stored specified via the SASMSTORE= option.

```
libname macrocmp "c:\sas\macros\compiled";
options mstored
        sasmstore = macrocmp;
```

The macro definition must specify the /STORE option and the macro code must be submitted to compile and store the macro.

```
%macro compiled_macro /store des = 'Meaningful macro description';

 %put Now executing compiled_macro ;

%mend;
```

When the contents of the SAS macro catalog are displayed, relevant information is provided, including the description specified when the macro was compiled and stored.

```
proc catalog catalog=macrocmp.sasmacr;
 contents;
run;
```

```
                        Contents of Catalog MACROCMP.SASMACR

# Name             Type          Create Date       Modified Date Description
ƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒƒ
1 COMPILED_MACRO   MACRO  12JUN2010:14:53:35  12JUN2010:14:53:35 Meaningful macro description
```

While the compiled macros are all in one location (ie. the permanent SASMACR catalog), the source code for those macros could be anywhere. It's strongly advised that one central repository be maintained for macro source code as well to ensure the source code can be located and enhanced when necessary.

AUTOCALL FACILITY

The AUTOCALL facility allows the availability of macro definitions to the SAS session via the setting of the MAUTOSOURCE option (on by default) and the specification of SASAUTOS directory or file references. File references can point to a directory or a specific source catalog. By default, the SAS configuration file (eg. sasv9.cfg) contain directory definitions for the AUTOCALL macros supplied with the various SAS products thus making them available to your SAS session, eg. %LEFT, %LOWCASE.

If user-written macros are to be included in the SASAUTOS search path, they must be added via the SASAUTOS= option. When SAS searches AUTOCALL locations, it will be in the order specified in the SASAUTOS option. As seen below, AUTOCALL specifications can include hard-coded paths, file references, SAS source catalog file references and the existing SASAUTOS definitions.

```
libname macros 'c:\sas\macros\sasautos\catalogs';    * location of source catalog;
filename catmacro catalog 'macros.macrocat';         * specific macro source catalog;
filename othmacro 'c:\sas\macros\sasautos\project'; * dir containing macro source;

options mautosource
        sasautos = ( catmacro othmacro "c:\sas\macros\sasautos" SASAUTOS );
```

When a macro is invoked but is not found in a SASMACR catalog (either WORK or the permanent library identified by SASMSTORE option), the AUTOCALL concatenation is searched. AUTOCALL definitions include directories containing .sas files or source catalogs, file references that point to directories containing .sas files or source catalogs, or file references that point directly to SAS catalogs (rather than the entire directory).

Macro source entries must bear the same name as the single macro code they contain. For example, if a macro called **%customer_profit** is to be defined in a Windows / *nix (ie. Unix or Linux) AUTOCALL library, the source code must be stored in a file named **customer_profit.sas** or in a catalog SOURCE entry named **customer_profit**. When a macro is found in an AUTOCALL location, the source code is retrieved and the macro compiled into WORK.SASMACR (or if /STORE has been specified and SASMSTORE set, the permanent location) making it available for execution.

MACRO SEARCH ORDER

How does SAS decide which libraries / locations to search when looking to execute a macro, and… in what order ? We have seen three distinctly different ways to make macros available to your SAS session. Since it's possible to (inadvertently) have a macro with the same name in each of these areas, the search order is important.

1. WORK.SASMACR
2. compiled stored macros
3. AUTOCALL locations

When a macro is referenced or invoked, the macro facility first looks in WORK.SASMACR for the compiled macro. If the macro is not found and the MSTORED option is on, the SASMACR catalog found in the permanent library identified by the SASMSTORE option is searched. As a last resort, if the MAUTOSOURCE option is set, the SASAUTOS locations are searched in the order that they are specified in the SASAUTOS specification for AUTOCALL macros.

If the macro source code is found in the AUTOCALL locations, then the macro is compiled to WORK.SASMACR and executed. If the MAUTOLOCDISPLAY option is on, the log will display the AUTOCALL location from which the macro source was retrieved. Since subsequent invocations of the now compiled AUTOCALL macro will be found in WORK.SASMACR, it will not be compiled again from the AUTOCALL location even if the macro source has been changed in the interim.

WHICH MACROS ARE AVAILABLE TO MY SAS SESSION?

The rest of the paper will discuss the author's **%list_sasautos** utility macro. **%list_sasautos** examines each of the locations where SAS macros may be defined and retrieves the macro names available from each area, creating a SAS dataset of the results and outputting them in order by macro name and search order. While sections of the macro will be discussed in turn here, the entire macro code may be found in the Appendix.

MACRO DEFINITION

```
%macro list_sasautos(help,work=Y);
```

The **%list_sasautos** macro has one positional parameter, **help**, and one keyword parameter, **work**. The positional parameter **help** is optional and only comes into play when the user executes the macro specifying a value of **?** or **HELP** to produce the macro generated documentation. See below.

```
3143  %list_sasautos(?);

%list_sasautos(work=Y);

Lists the .sas files and catalog source/macro objects found within directories surfaced by:
   - getoption('sasautos')            - SASAUTOS altered by options statement
   - pathname('sasautos')             - config file SASAUTOS definition
   - filerefs / catalogs found within SASAUTOS definitions
   - pathname(getoption('sasmstore')) - compiled macros.

If &work=Y ( default ), compiled macros from the WORK library will be included as well.

NOTE: Not every .sas / source module found within these directories is
NOTE: NECESSARILY a macro definition.  %list_sasautos does NOT open up
NOTE: objects to verify the presence of the required %macro statement.

In addition to the OUTPUT report ( use ODS for fancier report formats ),
results are also available in the WORK._LIST_SASAUTOS dataset.
```

As the help text indicates, if the **work** keyword parameter is not specified, compiled macros from the work directory will be included in the output. To omit macros from the WORK.SASMACR catalog, specify work=N.

DEFINING THE ENVIRONMENT

If the macro &work parameter indicates the WORK.SASMACR catalog is to be included and the WORK.SASMACR catalog exists, the work directory path is retrieved and enclosed in single quotes. In addition, if the SASMSTORE option is set, the quoted path to the compiled stored macro catalog is accessed and saved in a macro variable for later use (see Processing SAS Catalog Entries below).

```
/*
   work.sasmacr and the sasmacr catalog found at the SASMSTORE location
   are the first searched.  If &WORK=Y, grab the path info for the work
   directory.  If the SASMSTORE option is set, surface the path information
   for that library.  We'll include at the front of the concatenation since
   reflects the search order SAS uses.
*/

%if %upcase(&work) = Y and %sysfunc(cexist(work.sasmacr)) %then
   %let work_lib = %unquote(%str(%')%sysfunc(pathname(work))%str(%'));
%else
   %local work_lib;

%let sasmstore = %sysfunc(getoption(sasmstore));
%if &sasmstore ne %then %do;
   %let sasmstore = %sysfunc(pathname(&sasmstore));
   %if &sasmstore ne %then
      %let sasmstore = %unquote(%str(%')&sasmstore%str(%'));
%end;
```

DATA STEP INITIATION

For source catalog definitions, the **%list_sasautos** macro constructs the full path to the catalog. Since it may be invoked in both Windows and *nix environments, a variable, **s**, is defined with the environment appropriate slash.

```
data _list_sasautos ( keep = path member order type catalog );

   if substr(upcase("&sysscp"),1,3) = 'WIN' then
      s = '\';
```

```
else
   s = '/';

length pathname
      sasautos  $32000
      path
      chunk
      option  $2000
      member  $200
      catalog   $32
      type    $8
      ;

label path    = 'O/S Path'
     member      = 'Macro / Filename'
     type        = 'Object Type'
     catalog     = 'Catalog Name'
     order    = 'Resolution Order'
     ;
```

PARSING SASAUTOS OPTION SETTING

Once the SASAUTOS option value is retrieved, the macro parses the path / file reference components, extracting the salient bits and retrieving the path / catalog information where required. Note that the code is making use of the SCANQ function that ignores separator characters within quotes. This allows the macro to support Windows pathnames containing spaces.

If the SASAUTOS specification is a file reference (including the “SASAUTOS” file reference), the PATHNAME function retrieves the actual path allocated to the file reference. Since file references can also point to specific catalogs, where that is the case, the path and catalog name is included in the fleshed out SASAUTOS concatenation being built.

When this section completes, the SASAUTOS variable only contains directory paths and fully qualified catalog names. All file references have been resolved. Work library and the compiled stored macro locations are pre-pended to the SASAUTOS paths if available.

```
option      = compress(getoption('sasautos'),'()');      /* get SASAUTOS option value     */

/*
   Grab space-delimited SASAUTOS definitions.  Entries that are file system paths will be
   captured and file references, SASAUTOS and catalog references will be expanded.
*/

do I = 1 by 1 until ( scanq(option,I,' ') = ' ' );
   chunk = compress(scanq(option,I,' '),"'");

   if indexc(chunk,':/\') then do;        /* if path delimiters found, pass straight in  */

      sasautos             = catx(' ', sasautos, chunk );

   end; else do;                 /* no path delimiters found, expand entry to path level  */

      pathname = compress(pathname(chunk),'()');
      if pathname > ' ' then do;

         if pathname =: '..' then do;           /* catalog libname starts with ..  */
            cat_pathname   = substr(pathname,3);   /*  skip over two dots              */
            /*  Since we have catalog name, enclose with single quotes and insert in path  */

            path        = pathname(scan(trim(cat_pathname),1,'.')) || s ||
                          scan(trim(cat_pathname),2,'.') || '.SAS7BCAT';
            sasautos       = catx(' ', sasautos, "'"||trim(path)||"'" );

         end; else do;  * have SASAUTOS or a fileref, SASAUTOS paths start with single quote ;

            if left(pathname) =: "'" then
               sasautos    = catx(' ', sasautos, compress(pathname,'()'));    /* sasautos */
```

```
            else
               sasautos    = catx(' ', sasautos, "'"||trim(pathname)||"'");   /* fileref  */
         end;
      end;
   end;

end;

/*
   If we're going after WORK and COMPILED STORED macros, add their paths at
   the front since that's the search order SAS uses
*/

sasautos = left("&work_lib &sasmstore " || translate(sasautos,"'",'"'));

cat   = 0;
order    = 0;
```

RETRIEVAL OF MACRO MODULES

The long section of code that follows appears somewhat involved because of the multiple steps required to traverse the SASAUTOS directories and retrieve the files within each. However, when it's broken down, the steps are not complex.

1. paths determined in the Parsing SASAUTOS Options Setting are extracted one by one
2. if the path indicates a specific SAS macro catalog, the catalog routine is executed to save the catalog data for additional processing
3. if the path does not indicate a SAS macro catalog, the directory is opened and the number of files in the directory determined
4. the directory's file names are read
    a. if the file is a SAS catalog, the catalog routine is executed to save the catalog data for additional processing
    b. if the file is a .SAS file, an observation is created in the _list_sasautos dataset
    c. otherwise the file is ignored
5. opened files / directories are closed
6. once all paths have been examined, the number of catalogs to be dealt with is written to a macro variable
7. catalog procedure saves catalog name, path and the order it occurs in the SASAUTOS concatenation.

```
do i = 1 by 1 until ( scanq(sasautos,I,' ') = ' ' );

   path = compress(scanq(sasautos,i,' '),"'");
   /*
      Where the fileref pointed to a SAS catalog, we have specified the SAS7BCAT suffix to
      ensure we pick up only the specified catalog at this path.  Since we're processing
      catalogs in more than   one spot, we're using a common routine
   */

   if scan(path,-1,'.') = 'SAS7BCAT' then do;
      member   = scan(path,-1,s);
      path  = substr(path,1,length(path)-length(member)-1);
      link do_cat;      /* catalog identified via fileref                        */

   end; else do;     /* if it ain't a catalog, it must be a directory         */

      problem = filename('dir',path);     /* create a fileref pointing to dir    */

      if not(problem) then do;
         d = dopen('dir');             /* open the directory              */
         if d then do;                 /* directory successfully open?    */
            num = dnum(d);             /* number of files in directory  */

            do _i = 1 to num;          /* loop through files in directory                */
               member   = dread(d,_i); /* get next filename in directory                 */

               /*
                  Try to append the member name to the end of the path and
                  open it as a directory, if that's successful, we don't want it,
                  ie. we only want real files
```

```
                */

                dir_test_file  = filename('dir_test',cats(path,s,member));

                if dir_test_file then continue;     * cannot assign filename, iterate loop ;
                dtf = dopen('dir_test');
                if dtf then do;
                   rc = dclose(dtf);
                   continue;                  * file opened as a sub-directory, iterate loop ;
                end;

                if upcase(scan(member,-1,'.')) = 'SAS7BCAT' then do;

                   link do_cat;      /* found a catalog in the directory, deal with it   */

                end; else do;

                   if upcase(scan(member,-1,'.')) = 'SAS' then do;    /* .sas member ? */
                      order    + 1;
                      type  = '.sas';
                      output;
                   end;
                end;
             end;
             rc = dclose(d);              /* close the directory we've opened */
          end;
       end;
    end;
   end;

   call symputx('no_of_cats',cat);   /* save the no. of catalogs found for macro loop below  */

   return;

/*
   This code is specified here and LINKed to from two different places.  NOT using
   sashelp.vcatalog because it just got toooooo complicated because I don't have
   the libname.  Gets complicated with compiled macros.  We're getting catalog
   details in the following step.
*/

do_cat:
   cat   + 1;
   order + 1;
   call symput ('path'||put(cat,3.-l),trim(path));
   call symput ('cat'||put(cat,3.-l),substr(member,1,length(member)-9)); * take off .sas7bcat;
   call symputx('order'||put(cat,3.-l),order);
return;

run;
```

PROCESS SAS CATALOG ENTRIES

For each catalog found in the previous step, extract the applicable contents data and append it to the _list_sasautos dataset. We're only interested in SOURCE and MACRO catalog entries.

```
/*
   Now unpack the contents of the macro/source catalogs, working our way
   through the macro variables created in the previous step.  Routing
   source/macro contents of the catalog to an OUT dataset.
*/

%do i = 1 %to &no_of_cats;

   libname _c &&path&i;

   proc catalog catalog = _c.&&cat&i;
      contents out = _cat_contents ( keep = memname name type
                           where = ( type in ( 'SOURCE', 'MACRO' ) ));
   quit;

   /*  Any catalog entries found that we're interested in?  */

   data _null_;
      if 0 then set _cat_contents nobs = nobs ;
      call symputx('_cat_contents_nobs',nobs);
      stop;
   run;

   %if &_cat_contents_nobs > 0 %then %do;

      /*  Flesh out the details  */

      data _cat_contents;
         set _cat_contents ( rename = ( name = member memname = catalog ));
         length path $2000;
         path  = "&&path&i";
         catalog  = "&&cat&i";
         order    = &&order&i;
         type  = lowcase(type);  * prefer lowercase ;
      run;

      /*  Append to those we've found already  */

      proc append base = _list_sasautos
               data = _cat_contents    force;
      run;

   %end;

   libname _c clear;

%end;
```

SORT AND REPORT

Since it is possible that the SASAUTOS specification could inadvertently include duplicate path information, sort the _list_sasautos dataset to remove the duplicates. Entries are first sorted by the order variable to ensure the earliest entries in the concatenation are retained.

For reporting purposes, the dataset is sorted by member name and order to ensure the concatenation order for macros with more than one definition are identified.

```
/*  It's possible that duplicate paths are in the SASAUTOS definition, weed out dups here  */

proc sort data = _list_sasautos;
   by path member order;
run;

/*  Sorted by "order" so we'll keep the earliest one found in concatenation  */

proc sort data = _list_sasautos  nodupkey ;
   by path member;
run;

/*  Sort the list by member name and the order the macro was found  */

proc sort data = _list_sasautos;
   by member order;
run;

proc print data=_list_sasautos noobs ;
   var order member path type catalog ;
   format _character_;
run;
```

CONCLUSION

Sharing modularized or utility SAS code as macros within an organization or project can be advantageous. Coding effort is reduced, business rules can be captured in one spot and programmer efficiency increased. Properly defined SAS configuration files or autoexec procedures will ensure users have the required access to those modules, whether they be via %INCLUDE of modules, compiled stored macros or macro code made available via an autocall facility. **%list_sasautos** can be very helpful in disclosing the shared modules available to your SAS session by examining the compiled stored macros and AUTOCALL specifications.

REFERENCES

Arthur L. Carpenter, "Building and Using Macro Libraries" SUGI27 Proceedings, http://www2.sas.com/proceedings/sugi27/p017-27.pdf. 2010-06-14.

Ronald Fehd, "A SASAutos Companion: Reusing Macros" SUGI30 Proceedings, http://www2.sas.com/proceedings/sugi30/267-30.pdf, 2010-06-14.

SAS Online Docs - HTTP://SUPPORT.SAS.COM/91DOC/DOCMAINPAGE.JSP

ACKNOWLEDGMENTS

Thanks to Marje Fecht, Laura House and Dianne Piaskoski for their proof-reading prowess. In spite of their heroic efforts, I'm sure errors remain for which I'll humbly take credit. **J**

CONTACT INFORMATION

The latest version of this paper and the source code for **%list_sasautos** is available from my website below. Your comments and questions (and corrections ?!?!) are valued and encouraged. Contact the author at:

Harry Droogendyk
Stratia Consulting Inc.
PO Box 145
Lynden, ON, L0R 1T0
905-296-3595
conf@stratia.ca
www.stratia.ca

APPENDIX - %LIST_SASAUTOS SOURCE CODE:

```
/*****************************************************************

      Macro:       list_sasautos;
      Author:      Harry Droogendyk        harry@stratia.ca
      Date:        2008/07/12

      Prints the .sas member names of the various directories and
      filerefs returned by the GETOPTION(SASAUTOS) and PATHNAME(SASAUTOS)
      functions.  Also examines WORK.SASMACR and the SASMACR catalog
      referenced by the PATHNAME(GETOPTION(SASMSTORE)).

      If the output is required in a file, use ODS or PROC PRINTTO to
      route the output appropriately.

      Parms:  work = Y to list WORK.SASMACR entries
      Output:  PROC PRINT

*****************************************************************/

%macro list_sasautos(help,work=Y);

%if &help = ? or %upcase(&help) = HELP %then %do;
      %put;
      %put;
      %put %nrstr(%list_sasautos(work=Y););
      %put;
      %put %nrstr(Lists the .sas files and catalog source/macro objects found within directories
surfaced by:);
     %put %nrstr(    - getoption('sasautos')        - SASAUTOS altered by options statement);
      %put %nrstr(   - pathname('sasautos')         - config file SASAUTOS definition);
     %put %nrstr(    - filerefs / catalogs found within SASAUTOS definitions);
     %put %nrstr(    - pathname(getoption('sasmstore'))    - compiled macros.);
     %put ;
     %put %nrstr(If &work=Y ( default ), compiled macros from the WORK library will be included
as well.);
     %put ;
     %put %nrstr(NOTE: Not every .sas / source module found within these directories is );
     %put %nrstr(NOTE: NECESSARILY a macro definition.  %list_sasautos does NOT open up );
     %put %nrstr(NOTE: objects to verify the presence of the required %macro statement.);
     %put ;
     %put %nrstr(In addition to the OUTPUT report ( use ODS for fancier report formats ), );
     %put %nrstr(results are also available in the WORK._LIST_SASAUTOS dataset.);
     %put;
     %goto exit;
%end;

/*
   work.sasmacr and the sasmacr catalog found at the SASMSTORE location
   are the first searched.  If &WORK=Y, grab the path info for the work
   directory.  If the SASMSTORE option is set, surface the path information
   for that library.  We'll include at the front of the concatenation since
   reflects the search order SAS uses.
*/

%if %upcase(&work) = Y and %sysfunc(cexist(work.sasmacr)) %then
   %let work_lib = %unquote(%str(%')%sysfunc(pathname(work))%str(%'));
%else
   %local work_lib;

%let sasmstore = %sysfunc(getoption(sasmstore));
%if &sasmstore ne %str() %then %do;
   %let sasmstore = %sysfunc(pathname(&sasmstore));
   %if %bquote(&sasmstore) ne %str() %then
      %let sasmstore = %unquote(%str(%')&sasmstore%str(%'));
%end;

data _list_sasautos ( keep = path member order type catalog );

   if substr(upcase("&sysscp"),1,3) = 'WIN' then
      s = '\';
```

```
else
   s = '/';

length pathname
      sasautos $32000
      path
      chunk
      option   $2000
      member   $200
      catalog  $32
      type     $8
      ;

label path        = 'O/S Path'
     member    = 'Macro / Filename'
     type         = 'Object Type'
     catalog      = 'Catalog Name'
     order        = 'Resolution Order'
     ;

catalog     = ' ';                              /* to avoid not initialized msg */
option      = compress(getoption('sasautos'),'()');      /* get SASAUTOS option value     */

/*
   Grab space-delimited SASAUTOS definitions.  Entries that are file system paths will be
   captured and file references, SASAUTOS and catalog references will be expanded.
*/

do i = 1 to &sysmaxlong until ( scanq(option,i,' ') = ' ' );
   chunk = compress(scanq(option,i,' '),"'");

   if indexc(chunk,':/\') then do;        /* if path delimiters found, pass straight in  */

      sasautos             = catx(' ', sasautos, chunk );

   end; else do;           /* no path delimiters found, expand the entry to the path level  */

      pathname = compress(pathname(chunk),'()');
      if pathname > ' ' then do;

         if pathname =: '..' then do;           /* catalog libname starts with ..  */
            cat_pathname   = substr(pathname,3);   /*  skip over two dots  */

            /*  Since we have catalog name, insert it in the path
                with surrounding single quotes  */

            path        = pathname(scan(trim(cat_pathname),1,'.')) || s ||
                          scan(trim(cat_pathname),2,'.') || '.SAS7BCAT';
            sasautos       = catx(' ', sasautos, "'"||trim(path)||"'" );

         end; else do;     * must have SASAUTOS or a file reference,
                             SASAUTOS paths start with a single quote ;

            if left(pathname) =: "'" then
               sasautos    = catx(' ', sasautos, compress(pathname,'()'));    /* sasautos */
            else
               sasautos    = catx(' ', sasautos, "'"||trim(pathname)||"'");   /* fileref  */
         end;
      end;
   end;
end;

/*
   If we're going after WORK and COMPILED STORED macros, add their paths at
   the front since that's the search order SAS uses
*/

sasautos = left("&work_lib &sasmstore " || translate(sasautos,"'",'"'));

put / 'Processing: ' option= // sasautos= ;

cat   = 0;
order    = 0;
```

```
   /*
      Chew through the fleshed out list of paths / catalogs.  We added quotes to
      paths in some cases so we could use the scanq below.  However, we want to
      remove them before we use the path
   */

   do i = 1 to &sysmaxlong until ( scanq(sasautos,i,' ') = ' ' );

      path = compress(scanq(sasautos,i,' '),"'");

      /*
         Where the fileref pointed to a SAS catalog, we have specified the SAS7BCAT suffix to
         ensure we pick up only the specified catalog at this path.  Since we're processing
         catalogs in more than   one spot, we're using a common routine
      */

      if scan(path,-1,'.') = 'SAS7BCAT' then do;

         member   = scan(path,-1,s);
         path  = substr(path,1,length(path)-length(member)-1);

         link do_cat;   /* catalog identified via fileref                  */

      end; else do;     /* if it ain't a catalog, it must be a directory      */

         problem = filename('dir',trim(path));     /* create a fileref pointing to dir */

         if problem then do;
            put 'Cannot open filename for ' path;
         end; else do;
            d = dopen('dir');          /* open the directory            */

            if d then do;              /* directory successsfully open?    */
               num = dnum(d);          /* number of files in directory  */

               do _i = 1 to num;          /* loop through files in directory              */
                  member   = dread(d,_i);    /* get next filename in directory            */

                  /*
                     Try to append the member name to the end of the path and open it
                     as a directory, if that's successful, we don't want it,
                     ie. we only want real files
                  */

                  dir_test_file  = filename('dir_test',cats(path,s,member));

                  if dir_test_file then continue;     * cannot assign filename, iterate loop ;
                  dtf = dopen('dir_test');
                  if dtf then do;
                     rc = dclose(dtf);
                     continue;                  * file opened as a sub-directory, iterate loop ;
                  end;

                  if upcase(scan(member,-1,'.')) = 'SAS7BCAT' then do;

                     link do_cat;      /* found a catalog in the directory, deal with it   */

                  end; else do;

                     if upcase(scan(member,-1,'.')) = 'SAS' then do;    /* .sas member ? */
                        order    + 1;
                        type  = '.sas';
                        output;
                     end;
                  end;
               end;
               rc = dclose(d);              /* close the directory we've opened */
            end;
         end;
      end;
   end;

   call symputx('no_of_cats',cat);   /* save the number of catalogs found for macro loop below
*/
```

```
   return;

/*
   This code is specified here and LINKed to from two different places.  NOT
   using sashelp.vcatalog because it just got toooooo complicated because I don't
   have the libname.  Gets complicated with compiled macros.  We're getting catalog
   details in the following step.
*/

do_cat:

   cat   + 1;
   order    + 1;
   call symput ('path'||put(cat,3.-l),trim(path));
   call symput ('cat'||put(cat,3.-l),substr(member,1,length(member)-9)); * take off .sas7bcat;
   call symputx('order'||put(cat,3.-l),order);

return;

run;

/*
   Now unpack the contents of the macro/source catalogs, working our way
   through the macro variables created in the previous step.  Routing
   source/macro contents of the catalog to an OUT dataset.
*/

%do i = 1 %to &no_of_cats;

   %put Processing catalog &&cat&i in &&path&i;

   libname _c "&&path&i";

   proc catalog catalog = _c.&&cat&i;
      contents out = _cat_contents ( keep = memname name type
                           where = ( type in ( 'SOURCE', 'MACRO' ) ));
   quit;

   /*  Any catalog entries found that we're interested in?  */

   data _null_;
      if 0 then set _cat_contents nobs = nobs ;
      call symputx('_cat_contents_nobs',nobs);
      stop;
   run;

   %if &_cat_contents_nobs > 0 %then %do;

      /*  Flesh out the details  */

      data _cat_contents;
         set _cat_contents ( rename = ( name = member memname = catalog ));
         length path $2000;
         path  = "&&path&i";
         catalog  = "&&cat&i";
         order    = &&order&i;
         type  = lowcase(type);  * prefer lowercase ;
      run;

      /*  Append to those we've found already  */

      proc append base = _list_sasautos
               data = _cat_contents    force;
      run;

   %end;

   libname _c clear;

%end;

/*  It's possible that duplicate paths are in the SASAUTOS definition, weed out dups here  */
```

```
proc sort data = _list_sasautos;
   by path member order;
run;

/*  Sorted by "order" so we'll keep the earliest one found in concatenation  */

proc sort data = _list_sasautos  nodupkey ;
   by path member;
run;

/*  Sort the list by member name and the order the macro was found  */

proc sort data = _list_sasautos;
   by member order;
run;

proc print data=_list_sasautos noobs ;
   var order member path type catalog ;
   format _character_;
run;

%exit:

%mend list_sasautos;
```