Globale Makrovariablen im Programmieralltag – aber sicher!

Renate Scheiner-Sparna
iOMEDICO AG
Hanferstraße 28
79108 Freiburg i.Br.
Renate.Scheiner-
Sparna@iomedico.com

Zusammenfassung

Globale Makrovariablen flexibel erzeugen, sicher platzieren, elegant verwenden und zuverlässig wieder löschen.
Dieser Beitrag wendet sich an alle, die an einer zusammenfassenden Darstellung der Erzeugung und Verwendung globaler Makrovariablen außerhalb von Makros interessiert sind, insbesondere auch an SAS Programmierer mit wenig oder ohne Erfahrung in Makro-Programmierung.
Auch ohne ein „alter Hase“ zu sein kann man die Funktionalität von Makrovariablen sicher anwenden und seine Programme effektiver und flexibler gestalten. Es werden einige konkrete Beispiele für die Anwendung erläutert. Neben der Syntax geht es auch um die Anwendungs-Sicherheit.

Schlüsselwörter: Makrovariable, Macro Facility, Sicherheit

1 Globale Makrovariablen und Anwendungs-Sicherheit

Globale Makrovariablen können sehr nützliche Hilfsmittel in der SAS Programmierung sein. Sie sind Informationsträger, die der Programmierer nicht ständig sieht, die aber große Auswirkungen haben können. Zudem kann sich ihr Inhalt immer wieder ändern. Deshalb ist der Aspekt der Anwendungs-Sicherheit von zentraler Bedeutung.
Programmier-Disziplin, Programmier-Standards und ein gewisser Respekt können vor leichtfertiger, fehleranfälliger Verwendung von Makrovariablen schützen. im Rahmen einer ausführlichen Betrachtung der Syntax zur Erzeugung, einigen Beispielen zur Verwendung und differenzierten Überlegungen zum Löschen von globalen Makrovariablen liegt hier der Schwerpunkt auf dem sicheren Umgang des Programmierers mit diesem Element.

2 Definition und zentrale Eigenschaften von Makrovariablen

Eine Makrovariable ist die Repräsentation einer Zeichenkette. Sie wird in SAS in einer Symbol-Tabelle gespeichert. Eine Makrovariable enthält immer Text.

Der Scope beschreibt den Gültigkeitsbereich einer Makrovariablen. Lokal bedeutet, dass die Gültigkeit einem Makro zugeordnet ist und die Makrovariable nur während der

Laufzeit dieses Makros zur Verfügung steht. Eine lokale Makrovariable ist in einer lokalen Symboltabelle gespeichert und ist nur während der Laufzeit des Makros gültig [1]. Global bedeutet, dass eine Makrovariable während der gesamten SAS-Sitzung zur Verfügung steht, falls sie nicht vorher aktiv gelöscht wird. Eine globale Makrovariable ist in der globalen Symboltabelle gespeichert.

Eine Makrovariable ist nicht permanent über eine SAS-Sitzung hinaus gültig. Eine Ausnahme bildet nur die Anwendung von `proc presenv`. Mit dieser Prozedur können Systemeinstellungen, Optionen und auch Makrovariablen für eine spätere Sitzung gespeichert werden.

Dieser Beitrag beschäftigt sich fast ausschließlich mit globalen Makrovariablen, da diese durch die prinzipiell unbegrenzte Gültigkeit besondere Aufmerksamkeit verdienen.

3 Arten von globalen Makrovariablen

Globale Makrovariablen können verschiedenen Ursprungs sein. Sie lassen sich unterteilen in System-Makrovariablen und nutzerdefinierte globale Makrovariablen.

3.1 System-Makrovariablen

System- Makrovariablen werden automatisch von SAS beim Start angelegt. Bis auf wenige Ausnahmen fangen ihre Namen mit SYS an. Sie enthalten Eigenschaften der aktuellen SAS-Sitzung, wie z.B.

- User
- Datum
- Systemeinstellungen
- SAS-Version

System-Makrovariablen können ins Log-Fenster ausgegeben werden mit folgender Syntax:

```
%put _automatic_;
```

Log-Fenster (Ausschnitt):

```
1    %put _automatic_;
AUTOMATIC AFDSID 0
AUTOMATIC AFDSNAME
AUTOMATIC AFLIB
AUTOMATIC AFSTR1
AUTOMATIC AFSTR2
AUTOMATIC FSPBDV
AUTOMATIC SYSADDRBITS 64
AUTOMATIC SYSBUFFR
AUTOMATIC SYSCC 0
AUTOMATIC SYSCHARWIDTH 1
```

```
AUTOMATIC SYSCMD
AUTOMATIC SYSDATASTEPPHASE
AUTOMATIC SYSDATE 23JAN17
AUTOMATIC SYSDATE9 23JAN2017
AUTOMATIC SYSDAY Monday
...
...
AUTOMATIC SYSSCP WIN
AUTOMATIC SYSSCPL X64_ES08R2
AUTOMATIC SYSSITE 70148145
AUTOMATIC SYSSIZEOFLONG 4
AUTOMATIC SYSSIZEOFPTR 8
AUTOMATIC SYSSIZEOFUNICODE 2
AUTOMATIC SYSSTARTID
AUTOMATIC SYSSTARTNAME
AUTOMATIC SYSTCPIPHOSTNAME SRV-SAS
AUTOMATIC SYSTIME 11:17
AUTOMATIC SYSTIMEZONE
AUTOMATIC SYSTIMEZONEIDENT
AUTOMATIC SYSTIMEZONEOFFSET 3600
AUTOMATIC SYSUSERID scheiner-sparna
AUTOMATIC SYSVER 9.4
AUTOMATIC SYSVLONG 9.04.01M3P062415
AUTOMATIC SYSVLONG4 9.04.01M3P06242015
AUTOMATIC SYSWARNINGTEXT
```

3.2 Nutzerdefinierte Makrovariablen

Im Gegensatz zu den System-Makrovariablen werden die nutzerdefinierten Makrovariablen vom User während der SAS-Sitzung angelegt und befüllt. Die verschiedenen Möglichkeiten dazu sind im Abschnitt 6 beschrieben. Hier exemplarisch eine einfache `%let` Anweisung:

```
%let userMV = KSFE2017;
```

Wenn diese Anweisung in SAS ausgeführt wurde, kann man sich die Makrovariable userMV und evtl. weitere vorhandene nutzerdefinierte Makrovariablen ins Log-Fenster ausgeben lassen mit:

```
%put _user_;
```

Dann erhält man folgende Log-Ausgabe:

```
3    %put _user_;
GLOBAL USERMV KSFE2017
```

4 Ausgabe von Makrovariablen ins Log-Fenster

Es gibt eine ganze Reihe von Befehlen, mit denen bestimmte Gruppen von Makrovariablen ins Log-Fenster ausgegeben werden können. Diese sind in der folgenden Tabelle zusammengestellt:

Tabelle 1: Ausgabe von Makrovariablen-Gruppen ins Log-Fenster

Syntax	Log-Ausgabe
`%put _automatic_;`	Alle System-Makrovariablen
`%put _user_;`	Alle nutzerdefinierten Makrovariablen
`%put _global_;`	Alle globalen Makrovariablen (System- + nutzerdefiniert)
`%put _local_;`	Alle lokalen Makrovariablen (immer nutzerdefiniert)
`%put _all_;`	Alle Makrovariablen (global und lokal)

Einzelne Makrovariablen können auf verschiedene Weise ins Log-Fenster geschrieben werden. Soll es der pure Makrovariablen-Inhalt sein, ein erklärender Text davor, die Einbettung in einen vollständigen Satz oder auch der Makrovariablen-Name gefolgt vom Inhalt: alles ist möglich. Die folgende Tabelle zeigt Beispiele für die Syntax zur Ausgabe der Makrovariablen SYSDATE9.

Tabelle 2: Ausgabe einer einzelnen Makrovariable ins Log-Fenster

Syntax	Log-Ausgabe
`%put &SYSDATE9.;`	23JAN2017
`%put System date: &SYSDATE9.;`	System date: 23JAN2017
`%put &=SYSDATE9;`	SYSDATE9=23JAN2017

5 Existenz einer Makrovariable

Ob eine Makrovariable aktuell in der SAS-Sitzung existiert, kann mit der Funktion `%symexist()` überprüft werden. Diese Information kann im Ablauf von Programmen weiterverwendet werden, wie z.B. in folgendem Data Step:

```
data class;
  set sashelp.class;
  if %symexist(userMV) then event1="&userMV";
  if %symexist(otherMV) then event2="&otherMV";
run;
```

Angenommen die Makrovariable userMV existiert, die Makrovariable otherMV jedoch nicht, erhalten wir folgendes Data Set:

VIEWTABLE: WORK.CLASS

	NAME	SEX	AGE	HEIGHT	WEIGHT	EVENT1	EVENT2
1	Alfred	M	14	69	112.5	KSFE2017	
2	Alice	F	13	56.5	84	KSFE2017	
3	Barbara	F	13	65.3	98	KSFE2017	
4	Carol	F	14	62.8	102.5	KSFE2017	

Abbildung 1: Work Data Set CLASS mit neuen Variablen EVENT1 und EVENT2 (Ausschnitt)

Im Hintergrund sucht SAS in den Symboltabellen nach der Makrovariablen und findet sie nicht. Das Log zeichnet deshalb hier eine Warnung auf:

```
WARNING: Apparent symbolic reference OTHERMV not resolved.
```

Es kann sinnvoll sein, diesen Data Step trotz Warning so zu verwenden, wenn der Programmierer bewusst mit der Makrovariablen umgeht und die Existenz / Nicht-Existenz als wichtige Information für seinen Work Flow wahrnimmt. Warnings dieser Art können dann verhindert werden, indem die Option NOSERROR gesetzt wird: `options NOSERROR;`
Hier handelt es sich jedoch um eine Option, die nicht generell gesetzt werden sollte. Zu groß ist die Gefahr, dass an anderer Stelle übersehen wird, wenn erwartete Makrovariablen nicht zur Verfügung stehen. Deshalb ist es hochgradig sicherheitsrelevant, diese Option unmittelbar nach dem bewussten Umsetzen und der Anwendung wieder zurückzusetzen auf SERROR. So kann man die Information über die Existenz einer Makrovariablen für die Programmierung nutzen, ein sauberes Log erzielen und mit Disziplin doch die Gefahr minimieren, Warnings ungewollt zu unterdrücken.

6 Globale Makrovariablen flexibel erzeugen

Außerhalb von einem Makro erzeugte Makrovariablen sind immer global gültig. Auch innerhalb von Makros kann man globale Makrovariablen erzeugen, das ist hier jedoch nicht Thema. Man sollte jedoch daran denken, dass diese Möglichkeit besteht und auch diese Makrovariablen im Blick behalten. Einen Überblick über die Methoden gibt Carpenter [2]. Leere globale Makrovariablen kann man mit `%global` anlegen, sie sind dann existent, haben aber keinen Eintrag.

6.1 Erzeugen von Makrovariablen mit %let

Obwohl Makrovariablen immer mit Text gefüllt sind, darf man sie nicht mit Data Set Character-Variablen verwechseln. Bei der Erzeugung oder Befüllen von Makrovariablen mit `%let` werden keine Hochkommata verwendet, außer sie sollen Teil des Inhalts werden. Die einfachste Syntax lautet z.B.: `%let weekday=Monday;`.
Standardmäßig werden keine führenden oder angehängten Blanks in die Makrovariable übernommen. `%let weekday=  Monday;` führt zum selben Ergebnis. Wenn Leerzeichen Bestandteil des Strings in der Makrovariable sein sollen, muss eine Quoting-Funktion verwendet werden: `%let weekday=%str(  Monday)`.

6.2 Erzeugen von Makrovariablen mit Proc SQL und select into

Proc SQL ist praktisch, um aus einem Data Set heraus Makrovariablen zu generieren [3], [4]. Hier sind sogar aggregierende Funktionen anwendbar, wie im folgenden Beispiel die Anzahl der Zeilen unter einer gegebenen where-Bedingung:

```
proc sql noprint;
  select count(*) into: NUMDEAD trimmed
  from sashelp.heart
  where status='Dead';
quit;
%put &=NUMDEAD;
```

Die Makrovariable heißt hier NUMDEAD, die Option `trimmed` verhindert führende Blanks. Im Log-Fenster erscheint:

```
NUMDEAD=1991
```

Ein weiteres Beispiel zeigt, wie man alle unterschiedlichen Einträge einer Variable Delimiter-getrennt in eine Makrovariable schreibt [5]. Die Anweisung `separated by` legt nicht nur den Delimiter fest, sondern verhindert auch führende Blanks.

```
proc sql noprint;
  select distinct DeathCause into: DCAUSE separated by '#'
  from sashelp.heart
  where status='Dead';
quit;
%put &= DCAUSE;
```

Log-Ausgabe:

```
DCAUSE=Cancer#Cerebral Vascular Disease#Coronary Heart
Disease#Other#Unknown
```

Die Anzahl dieser Einträge zählt sich einfacher mit `%let`:

```
%let NDCAUSE=%sysfunc(countw("&DCAUSE.", #));
%put &= NDCAUSE;
```

Log-Ausgabe:

```
NDCAUSE=5
```

6.3 Erzeugen von Makrovariablen im Data Step mit call symput / call symputx

Alternativ kann eine Makrovariable aus dem Data Step erzeugt werden. Die Makrovariable NDEAD wird in der letzten Zeile des Data Sets befüllt mit der internen Zeilennummer.

```
data _null_;
  set sashelp.heart (where=(status='Dead'))
                    end=eof;
  if eof then call symput('NDEAD', put(_n_, best.));
run;
```

Log-Ausgabe:

```
NDEAD=        1991
```

Mit call symputx lautet die Zeile:

```
  if eof then call symputx('ndead', _n_);
```

Log-Ausgabe:

```
NDEAD=1991
```

Die Unterschiede zwischen call symput und call symputx sind:

- Call symputx behält keine führenden Blanks.
- Es erscheint keine `converted to character values` Note im Log, falls es eine numerische Variable ist.
- die Länge für eingelesene numerische Werte ist 32 (anstatt 12 bei call symput)

6.4 Anwendungssicherheit bei der Erzeugung von Makrovariablen

Um die Übersicht über Makrovariablen zu behalten, werden sie idealerweise einmalig an zentraler Stelle erzeugt. Makrovariablen-Namen sollen nicht mehrfach für verschiedene Zwecke verwendet werden. Namens-Konventionen für Makrovariablen aus bestimmten Bereichen oder für bestimmte Zwecke sind an dieser Stelle ebenso hilfreich wie informative Kommentare. Die Inhalte werden einer Makrovariablen am Anfang der Programmsequenz zugewiesen, wo sie gebraucht werden, und am Ende dieser Programmsequenz wieder entfernt. Das alles klingt trivial, ist jedoch nur mit Disziplin und Programmier-Konventionen durchzuhalten.
Wenn eine Information als Inhalt einer Makrovariable vorliegt, sollte diese auch konsequent verwendet werden. Eine Mischung mit hartcodierter Information ist sehr fehleranfällig. Wird z.B. ein Programm angepasst für eine weitere Verwendung, kann es passieren, dass sich der Programmierer auf die Makrovariable verlässt und übersieht, wenn an einer Stelle hartcodierter Text anstelle der Makrovariablen verwendet wird.

Als Beispiel sei hier die Erzeugung einer Reihe von Outputs für mehrere Subgruppen genannt. Der Programmierer möchte die Makrovariablen nur einmal an zentraler Stelle belegen. Es gehört ebenso zur Anwendungssicherheit, keine Subgruppe zu vergessen.
Hier bewährt es sich, eine Liste aller Subgruppen einmalig in eine Makrovariable zu schreiben (hartcodiert wie hier gezeigt, oder automatisiert wie in Abschnitt 6.2 beschrieben) und nacheinander abzuarbeiten. Um diese Liste abzuarbeiten, gehen wir ganz kurz über den Focus dieses Artikels hinaus, denn hierfür ist eine Makro-Schleife das ideale Mittel. Eine Makro-Schleife kann nur innerhalb vom einem Makro aufgerufen werden.

Erstellen der Makrovariablen:

```
%let SUBGROUPS= Group 1#Group 2#Group 3;
%let NGR=3;
```

Kommentieren z.B. auch im Log:

```
%put These subgroups are processed: &= SUBGROUPS;
%put Number of subgroups: &=NGR;
```

Makro zum Abarbeiten der Subgruppen:

```
%macro groupselect(SUBGR=);

  %let nforLoop=%sysfunc(countw("&SUBGR.", #));
  %put Total number of loops: &=nforloop;

  %do i=1 %to &nforLoop;
```

```
    %put Loop &i.: ...Analysis code following...;
  %end;

%mend groupselect;
```

Der Makroaufruf in den beiden folgenden Zeilen ist gleichbedeutend, da sich hinter der Makrovariablen SUBGROUPS genau der String verbirgt, der in der anderen Variante hartcodiert mitgegeben wird.

```
%groupselect(SUBGR =%str(Group 1#Group 2#Group 3));
%groupselect(SUBGR =%str(&SUBGROUPS.));
```

Log-Ausgabe:

```
These subgroups are processed: SUBGROUPS=Group 1#Group 2#Group 3

Number of subgroups: NGR=3

Total number of loops: NFORLOOP=3
Loop 1: ...Analysis code following...
Loop 2: ...Analysis code following...
Loop 3: ...Analysis code following...
```

7 Globale Makrovariablen sicher platzieren

Um eine Sicherheit in der Platzierung von Makrovariablen zu erzielen ist es sinnvoll, Makrovariablen-Gruppen immer nach demselben Prinzip zu bilden und dabei Namenskonventionen zu verwenden. Das erleichtert die Wiedererkennung, die Weiterverwendung von Programmen und auch eventuelle Fehlersuche oder Validierungen. Nach dem folgenden Beispiel könnte z.B. eine Flag-Variable für die Abfrage aus dem Analyse-Data Set ADSL in MV1, ein zugehöriger „schöner“ Text für Labels und Titles in MV2 und ein Kurztext für eine Abfrage in einem anderen Analyse-Datensatz in MV3 stehen:

```
%let MV1=GROUPAFL;
%let MV2=%str(Group A: Bevacizumab xxx mg);
%let MV3=%str(BEVAXXX);
```

8 Globale Makrovariablen elegant verwenden

Die vielfachen Einsatzmöglichkeiten von Makrovariablen können im Rahmen dieses Beitrags nur kurz und punktuell dargestellt werden. Neben den bereits erläuterten Beispielen sind in Abschnitt 11 einige weitere Anregungen aufgeführt. Im Laufe der eigenen Programmierung werden sich immer wieder neue Anwendungen für den Einsatz globaler Makrovariablen ergeben.

9 Globale Makrovariablen zuverlässig löschen

Am Ende einer Programmsequenz wird in aller Regel ein Clean-up durchgeführt. Data Sets im Work-Verzeichnis zu löschen ist Standard, doch auch Makrovariablen müssen

bereinigt werden. Der Programmierer muss sich überlegen, wann eine Makrovariable nicht mehr gebraucht wird. Im Idealfall werden alle erzeugten Makrovariablen nach und nach wieder gelöscht. Praktisch gesehen kann man am Ende von jedem Programm einen Schritt zum Löschen vorsehen, der bei Bedarf befüllt wird.
Wenn man bereichsweise Makrovariablen löschen möchte, bewähren sich erneut Namenskonventionen. Sie ermöglichen es, selektiv Makrovariablengruppen zu löschen [6], wie wir in einem Beispiel sehen werden.
Die einfachste Form des Löschens erfolgt hartcodiert:

```
%symdel ngr subgroups;
%symdel ngr subgroups /nowarn;
```

Die Option `nowarn` verhindert eine Warning im Log, falls es eine Makrovariable nicht gibt, die in `%symdel` aufgeführt ist.

Möchte man eine Namenskonvention zum Löschen verwenden, hat man verschiedene Möglichkeiten. `Call symdel` und `Proc sql` mit `%symdel` werden hier gezeigt.

Angenommen wird, dass die folgenden Makrovariablen erzeugt wurden:

```
%global _userMV1 _userMV2 _userMV3 _userMV4;
%let _gPath=u:\&sysuserid.\;
%let _gDBDat=20160630;
```

Löschen mit Call Symdel:
Erzeugen eines Data Sets mit den Namen der Makrovariablen aus Meta-Information in SASHELP.VMACRO:

```
data macinfo2;
  set SASHELP.VMACRO;
  where name=:'_USER' or name=:'_G';
run;
```

Call symdel Aufruf:

```
data _null_;
  set macinfo2;
  call symdel(name);
run;
```

Löschen mit Proc SQL und %symdel:

```
proc sql noprint;
  select name into: ALLUSERMV separated by ' '
    from SASHELP.VMACRO
    where index(name, '_USER') or index(name, '_G');
quit;

%symdel &ALLUSERMV. ALLUSERMV;
```

Hier ist zu beachten, dass zuerst mit `select into` noch eine weitere Makrovariable ALLUSERMV erzeugt wird, in der die Namen der anderen Makrovariablen enthalten

sind. Diese muss daher auch beim Löschvorgang berücksichtigt und extra aufgeführt werden.

10 Globale Makrovariablen in Konflikt mit Makros

Abschließend zum Thema Sicherheit sei hier noch auf die Problematik hingewiesen, dass globale Makrovariablen ihren Inhalt ändern können, wenn in einem Makro eine Makrovariable gleichen Namens angesprochen wird [7]. Ein Programmierer, der das Makro nur verwendet und nicht im Detail kennt sucht unter Umständen lange, um einen solchen Fehler zu finden. Hier ist besondere Aufmerksamkeit notwendig. Namenskonventionen, die bestimmte Namen für Makrovariablen vorgeben, sind an dieser Stelle eine große Hilfe.

11 Beispiele für pfiffige Verwendungen von globalen Makrovariablen

11.1 SYSLAST Key

In der System-Makrovariable SYSLAST verbirgt sich immer der Name des zuletzt erzeugten SAS Data Sets. Wenn man als Programmierer während der Entwicklung von Programmen gern seine Schritte in den erzeugten Data Sets nachvollzieht, lohnt es sich, diese Information in einen Display Manager Key zu schreiben. Diese findet man, wenn man mit aktivem Programm-, Output- oder Log-Fenster den Befehl keys in der Kommandozeile absetzt. Alternativ ist das Fenster über die Menü-Punkte Tools – Options – Keys zu öffnen. vt ist der Befehl für das Öffnen eines Viewtables. Der für F3 angezeigte Text kann hier eingetragen werden.

KEYS <DMKEYS>

Key	Definition
F1	help
F2	reshow
F3	vt "&syslast."

Abbildung 2: Key Definition für SYSLAST

Nun kann mit der Funktionstaste das Data Set geöffnet werden. Wenn dann noch der Befehl winclose in den Viewtable-Keys auf der benachbarten Funktionstaste liegt, erspart man sich viel Zeit.

11.2 Standardisierung von Titles und Footnotes

Dieses Beispiel ist vermutlich vielen nicht unbekannt und doch immer wieder erwähnenswert. Title und Footnote Statements können sehr gut aus einer ganzen Reihe von automatischen und nutzerdefinierten Makrovariablen befüllt werden.
die Routine `%sysget(SAS_EXECFILENAME)` ruft den Namen des Programms auf, das aktuell abläuft. Mit diesem Beispiel werden zusätzlich das Datum des Datenstands, der User und das Ablauf-Datum in der Fußnote dokumentiert.

```
%let ProgramName=%sysget(SAS_EXECFILENAME);
%put &=ProgramName;
footnote5 j=l "Program: &ProgramName."
          j=c "Database date: &gDBDat., User: &SYSUSER."
          j=r "Run date: &SYSDATE9.";
```

11.3 Fehler-Texte auslesen und verwenden

Der jeweils letzte Fehler, der im Programmlauf auftrat, wird in der System-Makrovariable SYSERRORTEXT gespeichert, analog die letzte Warning in SYSWARNINGTEXT. Diese Information kann verwendet werden, um z.B. nachfolgende Prozeduren zu stoppen, wenn ein Fehler auftrat oder den Programmablauf nach einer bestimmten Warning in eine andere Richtung zu lenken.

12 Schlusswort

Am Anfang angestellte, konzeptionelle Überlegungen kosten Zeit. Während der Programmierung aufgewandte Sorgfalt ebenfalls. Am Ende kann man sicher sein, dass sich dieser Aufwand auszahlt. Weniger Fehler, mehr Übersicht, wiederverwendbare Programme belohnen dafür.

Literatur

[1] Lokal vs. global symbol tables: https://sassoftware.wordpress.com/2011/03/30/understanding-sas-macro-symbol-tables/

[2] Carpenter, AL: Five Ways to Create Macro Variables. Paper HW03_05 https://www.coursehero.com/file/15025434/HW03-05PDF/

[3] Abolafia, J et al.: What Would I Do Without PROC SQL and the Macro Language. Paper 031-30 http://www2.sas.com/proceedings/sugi30/031-30.pdf

[4] Carpenter, AL: Storing and Using a List of Values in a Macro Variable. Paper 028-30 http://www2.sas.com/proceedings/sugi30/028-30.pdf

[5] Proc SQL and macro facility: http://support.sas.com/documentation/cdl/en/sqlproc/62086/HTML/default/viewer.htm#a001360983.htm

[6] Delete global MV http://sascommunity.org/wiki/Deleting_global_macro_variables

[7] Macro variables in nested macros http://support.sas.com/documentation/cdl/en/mcrolref/62978/HTML/default/viewer.htm#p1b76sxg9dbcyrn1l5age5j5nvgw.htm