Paper 037-2007

# Back Up with Each Submit and Save Your Sanity!

Suzanne Humphreys, Cardiome Pharma Corp., Vancouver, BC (Canada)


## ABSTRACT

Do you work in a small company and/or on a stand alone computer? No network back ups to rely on? Have you ever lost work when the system crashes or you accidentally overwrite a SAS® program? This paper will present a handy, easy to implement tool for maintaining ten rolling SAS program back up files. As you submit code in the enhanced editor window, the whole contents of the window are sequentially saved in a back up folder, even if you have started a new session of SAS. This enables you to be able to retrieve the whole program for the last ten 'submits'. Give yourself one less frustration during your working day! Intended audience – intermediate to advanced, Windows platform, based on SAS 9.1.3, interactive SAS.


**KEYWORDS:** back up files, PIPE device-type, FILENAME statement, DOS commands, DM commands

## INTRODUCTION

Local back up files are useful, you can retrieve recently deleted or overwritten files even if you are working on a network and if you have your own back up file it is often much quicker and easier to restore a lost program or fragment of code. I have been working with a back up file of the current contents of my enhanced editor or program editor window for several years after a colleague at a previous company introduced me to the commands required.

His suggestion was to use a command set to a shortcut key in the KEYS window so that a back up file was created with each 'submit'.

- Create the folder *c:\backup* on your *c:* drive.
- Type KEYS on the command line in your SAS session.
- Chose a free shortcut key (mine is F11).
- Type the following commands: `clear log; clear out; file 'c:\backup\lastprog.sas' replace; submit;`
- Close the KEYS window and save your settings.
- Submit your program or fragment of SAS code using your new shortcut key.

Using this shortcut key to submit the contents of your enhanced editor or program editor window results in the *lastprog.sas* file being created or recreated and containing ALL of the contents of your window even if you only submitted a fragment of code. The log and output windows are cleared too. This has been a very valuable tool over the years and has frequently saved me from rewriting code or picking fragments out of a log file. However, this method has limitations. After a SAS session has crashed and you need to retrieve your current code from the back up file, if you inadvertently run a set up or autoexec program you will have overwritten the lost code.

## ROLLING BACK UP FILE SYSTEM

Recently, after again overwriting a back up file, I began to think of ways to improve the current process. I decided what I really needed was to be able to save more then one file – keep a rolling ten back up files for example. Recently I stumbled across a piece of code using the PIPE device-type with a FILENAME statement and this became the seed of an idea for keeping sequential recovery files.

**Assumptions:**

- *c:\backup* folder exists
- You use interactive SAS in display manager (DM) mode on a windows platform
- You use the enhanced editor window for programming (the code can be amended if you use the program editor window)
- The SAS program *bprog.sas* is saved in *c:\backup* folder, *bprog.sas* will contain the code and commands for producing the rolling ten back up files

**Keys:**

- Type KEYS on the command line in SAS
- Chose an unused shortcut key
- Type the following commands: `pgm; inc 'c:\backup\bprog.sas'; submit;`

These commands result in moving from the enhanced editor window to the program editor window, including *bprog.sas* in the program editor window and submitting the program.

**Code:**

The following code is contained within *bprog.sas*:

```
/* bprog.sas */
/* SH 13Sep06 */
/* Create a rolling count of 10 backup files */

/* set macro var bck to 1, if there are no backup files present 1 will be created */

%let bck=1;
```

The macro variable *bck* will be used to determine which number of back up file is created in the sequence of 1-10. In case no back up files are currently in existence, this variable is set to 1.

```
/* bring in filename and date modified information from the backup directory */
/* keep only backup information */
/* /t:w time, last written /a:-d file info, not directories /OD order, oldest file
first */

filename backup pipe 'dir C:\backup /t:w /a:-d /OD';
```

The FILENAME statement is used with the PIPE device-type to access the file information from the *c:\backup* directory using DOS dir commands. If your file path contains spaces, the file path can be contained within double quotation marks. This step retrieves the time each file in the directory was last written and orders the file information with the oldest modified file first.

```
data backup(keep=fname orda);
   infile backup missover pad length=len;
   input @01 line $varying200. len;
  if index(upcase(line),' BACKUP');
  fname=input('B'||scan(compress(upcase(line)),2,'B'),$20.);
  orda+1;
run;
```

This step creates the data set *backup* from the file *backup* named in the FILENAME statement. The *backup* data set uses an input statement to grab all the file information – the variable FNAME picks out the filenames and only those containing the text string BACKUP are selected. An order variable, ORDA, is created as a sequential index variable to maintain the order of oldest modified file first.

```
/* find maximum backup file number (up to 10) */

proc sql;
 create table backup1
 as select (select max(input(reverse(scan(reverse(fname),2,'.P')),3.))
    from backup) as maxnum, a.*
     from backup a;
quit;
```

This step uses PROC SQL (other methods could be used) to determine how many back up files already exist and creates the data set (or table) *backup1*. I have set my maximum number of back up files as ten but any number could be used.

```
/* create macro var bck, if <10 backup files then set var to the last backup+1
if 10 backups are present then bck = the number of the oldest modified file */

data _null_;
 set backup1 (obs=1);
  if maxnum lt 10 then call symput('bck',compress(put(maxnum+1,2.)));
   else call symput('bck',compress(reverse(scan(reverse(fname),2,'.P'))));
run;
```

The macro variable *bck* is updated to contain the number of the back up file that should be created. The aim is to overwrite the oldest back up file, therefore if less than ten back up files exist, the next numbered file in the sequence needs to be created. If ten files already exist then the oldest modified file should be overwritten and this filename will be held in the first observation of the *backup1* data set.

```
/* delete data sets created */

proc datasets lib=work;
delete backup backup1;
quit;
run;
```

The data sets *backup* and *backup1* are deleted as they are no longer required.

```
/* run dm commands: this prog run from pgm so go back to wpgm, save present prog as
backup, delete macro var, resize, clear output, shut the program editor window,
clear log, submit program */

dm 'wpgm; file "c:\backup\backup&bck..sas" replace; %symdel bck; resi; cle out; pgm
off' editor;
** Submitting program **; dm 'cle log; sub;';
```

DM (display manager) commands control the windows (program editor, enhanced editor, log, output) and their contents. They are usually entered on the command line but can also be run from the enhanced editor or program editor window by preceding the commands with a DM statement and containing the commands within single quotation marks.

| | | |
|---|---|---|
| WPGM | - | moves to the enhanced editor window |
| FILE | - | saves the contents of the current window to the specified file |
| `%symdel bck;` | - | SAS code to delete the *bck* macro variable as this is no longer required |
| RESI | - | abbreviated command for RESIZE, will return the enhanced editor window to your standard sizing |
| CLE OUT | - | abbreviated command for CLEAR OUT, clears the output window |
| PGM OFF | - | shuts the program editor window as this is no longer required |
| EDITOR | - | ensures the program editor window closes as it is opened when commands are run from the enhanced editor window |
| CLE LOG | - | abbreviated command for CLEAR LOG, clears the log window |
| SUB | - | abbreviated command for SUBMIT, submits the contents of the enhanced editor window |

**Limitations:**

- The last line of code will appear in your log:
  `** Submitting program **; dm 'cle log; sub;';`
- The first run is slower (due to the PIPE device-type on the filename being initialized?)
- The pgm window will briefly open when you 'submit'
- You have to use 'save as' instead of 'save' to regularly save your program, using 'save' will result in just updating the back up file
- You cannot run procedures without specifying the data set name – you will get the following message in your log – "data set backup1 does not exist" as this will be the last created data set
- You cannot call any of your working data sets *backup* or *backup1*

**Advantages:**

- Never lose a program again. (As long as you do not submit another ten pieces of code before you realize!).
- Customize this code to your own preferences. For example, create more backup files, save them in a different directory, add other commands.

## CONCLUSION

However careful you are, mistakes happen and files are lost. It can be a lifesaver to keep back up files, especially as more errors tend to be made when you are rushed and under pressure. The PIPE device-type that can be used in conjunction with the FILENAME statement is a very effective tool that has many applications. In addition, DM commands have been around for years and yet are generally poorly understood yet extremely powerful. Documentation and examples are scarce but if you spend some time experimenting you can achieve the results you desire.

## REFERENCES

SAS Institute Inc. 2006. SAS technical support website. "Sample 1674: Dynamically determine the creation date and last modified date for an external file".
<http://support.sas.com/ctx/samples/index.jsp?sid=1674&tab=code>

Wikipedia website. "dir (DOS commands)".
<http://en.wikipedia.org/wiki/Dir_(DOS_command)>

SAS discussion: John Hendrickx 2005. "Open file in Textpad with SAS".
<http://listserv.uga.edu/cgi-bin/wa?A2=ind0510e&L=sas-l&O=D&P=4597>

SAS Institute Inc. 2003. SAS technical support website. "DM statements 'active window' arguments default to Program Editor window".
<http://support.sas.com/techsup/unotes/SN/005/005644.html>

SAS Institute Inc. 2001. SAS technical support website. "Sample 157: Reading multiple files with PROC IMPORT".
<http://support.sas.com/ctx/samples/index.jsp?sid=157&tab=code>

Wang, Hongwei. 2002. Rutgers University, NJ. "Utilizing the SAS Window Interface".
<http://www.stat.rutgers.edu/~hwang/WindowInterface.htm>

## ACKNOWLEDGMENTS

The author wishes to acknowledge Dan Higgins, whose original piece of code prompted the idea for this paper. Also, thanks to Stephen Hunt who gave his time to review and give valuable feedback.

## CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Suzanne Humphreys
Cardiome Pharma Corp.
6190 Agronomy Road, 6th Floor, Vancouver, BC, V6T 1Z3, Canada
Work Phone: 604 677 6905 ext. 314
Fax: 604 677 6915
E-mail: suzanne.humphreys@gmail.com