Unlimiting a Limited Macro Environment
David S. Septoff, SAS Consulting Services Inc., Rockville, MD

ABSTRACT
The full macro facility provides SAS® users with an extremely powerful programming tool. It allows for conditional execution of DATA and of PROC steps, creation of parameter-driven applications, passing SAS variables across steps, and much more. Unfortunately, the full power of the macro facility is not available in some operating environments. These programmers need not despair. SAS code can be written to simulate SAS macros using base SAS software available on all operating systems.

This paper presents an example of a SAS macro-based system developed in a VAX® environment and draws some conclusions on how SAS software, used as a programming language, can be just as powerful in a partial macro environment as it is in a full macro environment.

INTRODUCTION
The SAS System has experienced phenomenal growth in power, usage, and availability since its introduction. SAS software was originally designed to provide scientists and engineers with a data analysis tool that was powerful yet easier to use than traditional programming languages. The SAS DATA step gives users the data manipulation capability of a high level language, while many SAS procedures provide a wide variety of easy-to-use data handling, analysis and reporting capabilities. Unfortunately, the full power of SAS software is not available in all operating environments. Those that use the DEC® (Digital Equipment Corporation) version of SAS software, and all related environments, do not have the full power of the SAS macro facility. (SAS Institute announced at SUGI 13 that full macro will be available with the release of Version 8 for the minicomputer.) These SAS users should not despair. From this paper you will get, if nothing else, a generic approach, using base SAS software with its macro capabilities, to simulate the capabilities, of macro in the full macro environment.

This paper is designed to show how, in a partial macro environment, a SAS user can have power similar to a full macro environment. To explore this area, SAS processing as a whole must be placed in its proper context. The flow of the archetypical SAS job is shown in the following figure:

```
DATA
PROCEDURE
DATA
PROCEDURE
```

Note that each DATA and PROC step is compiled and executed separately and in sequence. The next figure shows the structure of a typical modular system, in which a main routine controls system execution and calls the appropriate subroutines:

```
MAIN
ROUTINE

SUBROUTINE
SUBROUTINE
SUBROUTINE

SUBROUTINE
SUBROUTINE
SUBROUTINE
```

The table below lists the three major requirements of software systems, and allows one to compare the SAS System with traditional programming languages as to the ability of each to satisfy the three requirements:

<table>
<thead>
<tr>
<th>System Requirements</th>
<th>Feature of Software Systems</th>
<th>SAS Macro Components of System</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modular Structure</td>
<td>A main routine governs execution and calls subroutines when needed. Main subroutine performs a particular function.</td>
<td>Macro which packages code into main and subroutines.</td>
</tr>
<tr>
<td>Conditional Execution</td>
<td>Procedures are executed only when required.</td>
<td>Code for conditional execution.</td>
</tr>
<tr>
<td>Parameter Control</td>
<td>Parameters governing an individual run of a system are passed to the system.</td>
<td>Macro variables.</td>
</tr>
</tbody>
</table>

One may look at the table above and be confused as to how, in a partial macro environment, could a system be developed using the same set of requirements. The table below may begin to answer this question:

<table>
<thead>
<tr>
<th>System Requirements</th>
<th>Solution in a Non-mainframe Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>Modular Structure</td>
<td>Files which package code into main routines and into functional pieces of the system.</td>
</tr>
<tr>
<td>Conditional Execution</td>
<td>DATA <em>NULL</em>, _UPD, _FILE, and _ INCLUDE statements.</td>
</tr>
<tr>
<td>Parameter Control</td>
<td>Parameter files and macro variables.</td>
</tr>
</tbody>
</table>
The requirement of modular structure, where a main routine controls the execution of the system and calls in subroutines, is different from standard SAS structure. It is this modular aspect of systems which is the most difficult for the SAS System to emulate; unfortunately, the other features of software systems are somewhat derived from this feature. For example, defining system execution parameters is a natural function of modular structure. In the SAS System without macro there is no easy way for a system to read parameter values where they are needed. Conditional execution of blocks of code is likewise a natural function of the modular structure. It is certainly not a natural function of the SAS System, in which each SAS step is separate and sequentially executed.

Macro, in a full macro environment allows a system based on the SAS System to have a modular structure in which a main macro can control execution of the SAS steps, calling in other macros as needed. In a partial macro environment, this does not have to change. Instead of a main macro, the system is developed using a main routine which will call subroutines and subroutines which will call other subroutines and so on. While macros allow for the packaging of code into functional units, so too do the subroutines just spoken of.

The only thing that differs is that the %MACRO-%MEND are not used within that partial macro environment. The routines, like macro, can contain anything from a DATA or PROC step to multiple DATA and/or PROC steps, and can be invoked whenever needed.

Macro variables still are the controlling parameters; they are generated by the system and can be set by the user. Conditional execution is still a natural feature in a system developed under this environment of macro. The packaging of code and the flexibility allowed by macro is still retained.

MACRO-BASED SYSTEM WITHIN THE PARTIAL MACRO ENVIRONMENT

Main Routines

The important constant of any software system is its main routine—the routine or code that functions as the supervisor of the system. It is the logical core of the system. A main routine in this environment is a set of code that initiates and directs the execution of the subroutines within the system. Compare this to a main routine with an environment with full macro environment and you can see how a main routine may be-concerned with anything else. The developer has the flexibility afforded through the full macro environment because all the code required parameters. Once provided with the parameters, the routine will produce the exact run desired. The user need not be concerned with anything else. The developer has the flexibility afforded through the full macro environment because all the code is generic and conditional. The generic nature of the code means that the macro variables need only be specified within the parm file (or whatever medium chosen to enter the run parameters).

Subroutines

Looking at the code above, you can see how a main routine may call other routines. This allows standard sets of code to be stored
as subroutines and brought in whenever necessary. Storing code in this manner allows for execution of functional parts of a system and contributes to the modularity of the code. The code below illustrates a subroutine:

```plaintext
Data _Readin;
  Update Master
    By iIndex;
  Run;
```

You can certainly see that there really is not much difference between this figure and a macro created in a full macro environment. This reflects the desire to keep code modular as well as functional.

Parameter Control

Parameters that control this system can, as stated earlier, be referenced in several different ways. Global parameters can be set and default parameters can be stated. The important point here is that there is no loss of power when programming in this type of environment rather than a mainframe environment.

CONCLUSION

While the SAS System and the full macro facility are excellent tools for large scale system development, the minicomputer user sometimes feels slighted by the lack of functions they may have on their system. The SAS macro facility in the minicomputer environment has been shown to have the capabilities of SAS macro on the mainframe. It can be truly modular in structure, and can permit conditional execution of code and parameter passing without much problem. At the same time, the finely controllable nature of these systems is such that end-users in batch and interactive modes can have great control over their runs without ever dealing with actual code.

ACKNOWLEDGMENTS

The author wishes to acknowledge Fidia Pharmaceutical Corporation for their support during the writing of his paper.

The author may be contacted at

SAS Consulting Services Inc.
1700 Rockville Pike
Suite 330
Rockville, MD 20852

SAS and SAS/AF are registered trademarks of SAS Institute Inc., Cary, NC, USA.

DEC is a registered trademark of Digital Equipment Corp., USA.

VAX is a trademark of Digital Equipment Corp., USA.