

Dealing with Memory Constraints under SAS 6.04 for DOS and moving to 6.06 under OS/2

by Philip Hamm and Robert Butler

Hay Systems, Inc.

4301 N Fairfax Dr. Suite 500

Arlington, Va. 22203

(703) 841-0079

This paper will discuss some of the memory problems with SAS version 6.04 under DOS and how they can be overcome. The focus will be on a large scale AF and FSP based application. The discussion will then turn to the task of moving the application to an OS/2 platform running SAS version 6.06. The inherent advantages of the OS/2 operating system and release 6.06 will be discussed.

The DOS application. The DOS operating system was first developed over ten years ago for an eight bit processor and a maximum of 640K memory. Over the years the operating system has become more and more sophisticated incorporating 16 bit technology, but the 640K limit can not change due to the way DOS applications access RAM. Now there are three basic types of memory that can be utilized in DOS for running SAS 6.04 efficiently. Figure 1 illustrates the three types of memory on a 286, 386, or 486 PC running DOS.

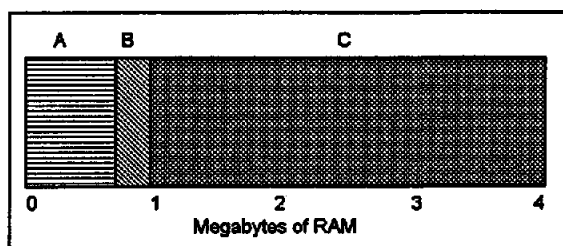


Figure 1

The section of the memory model labeled "A" represents the lower 640K base memory. SAS will use this space as it is available. Most SAS functions especially AF and FSP screens will only run in this space. Most DOS programs run here, as well as device drivers and TSR's (Terminate and Stay Resident programs, such as a mouse driver).

The section labeled "B" is the space between the 640K base and the end of the first megabyte. This is known as the "Upper Memory Block," or "High

Memory." Under DOS 5.0, this space can now contain most of the Operating System, as well as some device drivers and TSR's. Using the Upper Memory Block in this fashion can substantially increase the amount of free memory in the lower 640K, where SAS does most of its work.

The "C" section represents the remaining three megabytes of free RAM on this machine. This can be set up in a variety of ways. Since SAS uses *LIM expanded* RAM (also referred to as EMS), DOS 5.0's HIMEM.SYS extended memory driver can be used to run the EMM386.EXE Expanded memory driver (third party drivers such as 386MAX or QEMM386 could also be used). This will allow for the use of the expanded memory for SAS and the loading of device drivers and TSR's into the Upper Memory Block where they will not interfere with the workings of SAS 6.04. After these steps, all the available memory in the PC is optimized to work with SAS version 6.04. (NOTE: For the purpose of demonstration, there are three megabytes of expanded memory.)

The EMS command in the "config.sas" file determines how SAS will use available EMS memory. The "ALL" (-EMS ALL) parameter allocates 2048K, or two megabytes for SAS. The "ALL" really means "SOME", since only two megabytes will be accessed. If there is not sufficient EMS space, the option will be ignored. A "###" in this statement (-EMS ###) represents the number of 16K EMS pages which will be allocated for SAS to use. As with "-EMS ALL" the specified amount of EMS memory must be present, or the option will be ignored, and SAS will use only the base 640K. For example, in figure 1, there are three megabytes of EMS (All the extended memory in section "C" is configured as EMS). If the "-EMS ALL" parameter is used only two megabytes will be used by SAS. If the "ALL" is replaced with the number 192 (-EMS 192) three megabytes of expanded memory will be available for SAS. Due to variations between EMS

drivers, the physical amount of memory may differ from the logical amount.

While using this EMS trick to allow SAS to use more expanded memory does have its advantages, they are very limited. The reason is that there are a limited amount of SAS 6.04 modules which will use EMS memory. This means that even if you give SAS X megabytes of memory, it really will never be able to use more than about three. There is an advantage to using three instead of two megabytes, but more than that will make no difference.

As always, efficient programming will result in more efficient execution. This is especially true in SAS 6.04. Often efficient programming is necessary in PC SAS in order to avoid those nasty "Out of Memory" messages.

Saving memory in 6.04. Here are some specific examples of ways to conserve memory in SAS 6.04. Macros can be cleared from memory once they are used. This can reduce memory consumption especially when extensive macros are used. One macro "printtp" involved extensive macro processing, and therefore used a great deal of memory. The macro is defined and deleted as in Figure 2. When SAS runs, notes and source code (as well as many other optional debugging messages) are printed in the LOG window. In a fully interactive AF menu-driven system this LOG information is not useful except during development. The LOG notes and source can be turned off using the OPTIONS statement or window, or by starting the SAS session with the -NONEWS and -NONOTES switches.

```
%macro printtp;
--- Macro programming statements ---
%mend;
%printtp;
proc catalog cat = work.sasmacro et=macro ;
delete printtp;
run;
```

Figure 2

A system developed at Hay Systems, Inc. includes a data base which conforms to many Army standard codes. Users for whom the system is designed are not expected to memorize all of these codes, nor should they need to reference tables for their selections. All the codes which we use are included in a permanent format library and in data sets for the purpose of creating look-up tables.

Initially, we used FSEDIT sessions to facilitate these look-up tables, but found this method had some significant functional drawbacks. One was that different codes ended up in different data formats. A code with one hundred members was represented by a dataset with

ten observations of ten variables each. In contrast a smaller code only had only eight members, and was represented by a data set with eight observations. This type of variation forced us to make different FSEDIT selection screens for each code, which took up much hard disk space. A better option in this particular case was to use an AF extended table PROGRAM entry.

The benefits of using the extended table entry in this case are important. One benefit is that all the datasets now have the same format, with each member in a format represented by one observation. The code with one hundred members is now represented in a dataset with one hundred observations, just as the code with eight members is represented in eight observations. Another benefit is that now all the datasets representing the different codes can be accessed through one extended table. This is accomplished through the use of macro variables. Data set names are accessed in the AF PROGRAM through "CALL SYMGET" statements. Using one entry in a SAS catalog also conserves hard disk space.

An important subtlety in the interaction between SAS/AF and SAS/FSP and memory usage is the use of FSEDIT sessions within AF screens. The preferred method for this interaction is the seamless "CALL FSEDIT" SCL function. This is the preferred method for the user, since it requires no screen blinking and calls up the FSEDIT session quickly. The problem with SAS 6.04 and this method is that memory is not well used. While the FSEDIT session is in the foreground, the AF PROGRAM entry and all its memory overhead is taking space in the background. This is especially memory sensitive because neither the AF nor the FSP screen can be loaded into EMS memory. Both functions are stuck in the precious lower 640K. One solution is to use a SUBMIT CONTINUE block with a PROC FSEDIT. The PROC FSEDIT manages memory internally better than the CALL FSEDIT function. While this helps a little, the memory messages can persist. The only way to call a very large FSEDIT SCREEN entry through AF is to use a SUBMIT TERMINATE and PROC FSEDIT. The problem with this method is that the user sees the display manager. By turning all Display Manager windows and their borders, banners, etc. blue the user sees only a blue screen. Using %PUT statements in the code before the PROC allows the programmer to display messages to the user such as "Please Wait".

Even when using this method problems can arise with a very large FSEDIT screen containing much SCL code. Using PROC DISPLAYS for look up tables and with much data checking, the "Out of Memory" messages can persist. Another way to use memory more

effectively in SAS/FSP is to break up a large multiple page FSEDIT SCREEN into many small one page FSEDIT SCREEN entries called consecutively on the same data set.

Using AF MENU entries can indirectly cause memory problems. There are distinct advantages of using AF MENU entries. One advantage is that development is very quick and easy. Creating an application using MENU entries for branching control is very straightforward. The most significant specific example of this ease is the use of parent/child relationships. Using this MENU architecture makes MENU entries fast and backward branching easily managed. The problem with this method is in the way memory is used, specifically the lower 640K. AF MENU entries are loaded into the lower 640K of memory. This takes precious space needed for the operation of program functions such as PROGRAM entries and FSEDIT sessions where the actual work gets done. A more memory conservative approach is to use AF PROGRAM entries to emulate the functions of MENU entries. By using CALL GOTO routines with the 'C' parameter, one can create PROGRAM entries which function as menus. These PROGRAM entries can reside in memory one at a time, leaving much more space in the lower 640K for other SAS programs. Another advantage of using this method is cosmetic. Custom PROGRAM entries can be made uniform and neat; with some creative labels and attributes context sensitive help can easily be implemented. This can result in a much easier system for the end user.

"C" or some other lower level language can be used to control PC SAS as a main menu. Using SAS 6.04 can sometimes cause memory overload despite the most RAM efficient programming. Sometimes SAS uses RAM and does not free it. The low level language can be used to make a shell which calls up separate program modules. A "C" program which calls batch files can be used as a main menu. Using this method, a separate SAS 6.04 session can be initialized for each application "module".

Using some undocumented memory related commands often can help free some of the memory while working in SAS. The CDE command which has been outlined before in various technical reports at many SAS user group levels is a very useful development tool. When the "CDE" command is executed on the command line the SAS system performs a kind of memory inventory which displays a report in the Log window. There are variants and parameters which can be used with this tool. "CDM" or "CDE M" generates a report showing physical memory locations of various SAS

modules. The "CDE P" command purges unneeded SAS modules from memory leaving more room for SAS 6.04 to work. Using "CDE P" often in strategic situations can substantially reduce memory problems when using AF and FSP applications.

The OS/2 Application: This discussion will pertain to SAS 6.06 running under OS/2 version 2.0. The OS/2 operating system offers many advantages over the DOS environment. There are four major advantages of the OS/2 operating system compared to DOS for running SAS. The first and perhaps the single most important advantage is the use of a flat memory model. OS/2 2.0 has no 640K "limit" like DOS. The operating system treats all memory as one contiguous space. The second advantage is that OS/2 2.0 features a Graphical User Interface (GUI). This allows for push buttons, icons, mouse control, and other conveniences for the user. The third advantage is the use of preemptive multitasking and multithreading. Preemptive multitasking means that the CPU can run multiple programs at the same time. This means that if you wish to use SAS and PM Chart (A useful business charting tool bundled with OS/2) for example, you can. Both applications will share the processor. The advantage of multithreading, another OS/2 exclusive, is that one program can actually use the CPU to perform multiple internal tasks at the same time. SAS uses multithreading extensively, and the result is fast speed. The fourth advantage of OS/2 over DOS is the High Performance File System (HPFS) that OS/2 can use. This file system is much faster and more efficient than the DOS File Allocation Table system (Abbreviated FAT, this is the type of file system DOS uses exclusively. It was created for 360K and 180K diskettes over a decade ago). OS/2 can also read data from, and even be based on a FAT volume, but the HPFS is superior.

Future releases of SAS for OS/2 will take advantage of OS/2 2.0's 32 bit architecture. SAS 6.06 does not currently use this advanced design, but it can surely speed up an already fast system. SAS (as with any application) will run best on a computer when running under the most powerful operating system for that type of computer. The most sophisticated and powerful operating system for the Intel 80386, 80486, and compatibles is OS/2 2.0.

When moving from SAS 6.04 for DOS to 6.06 for OS/2 the first advantage is memory. All the memory problems discussed under the DOS system are gone. All the solutions are obsolete! All the SUBMIT TERMINATE edit sessions can be replaced by the much better CALL FSEDIT function. The "C" menu can be

replaced with a MENU or PROGRAM entry. All the time intensive "CDE P" statements can be deleted (carefully; the "CDE" statement is still here in 6.06). If the command lines have been colored the same color as the background in your AF application, you can turn them off. Leave the macros in RAM and use them many times. Make your many pages of FSEDIT on one data set one FSEDIT session again. Use MENU entries if you wish (Though PROGRAM entries are much more flexible and customizable).

Moving to 6.06. Moving to version 6.06 is a difficult process to anyone only familiar with the DOS release. The Display Manager is the first noticed difference. There are menu bars and window handles. Pop down a menu entry and a plethora of SAS options, windows, help, commands, etc. is immediately available. Minimize the LOG or OUTPUT, maximize the PROGRAM EDITOR, call help, install products, access utilities. There are many new tools available for the SAS user first discovering SAS running on a GUI. While there is a definite learning curve, generally SAS 6.06 under OS/2 is a much easier and flexible release to use than is the DOS version. After a short start up, most will find the release faster to both use and execute. SORTs and TABULATEs will be faster, as well as virtually all other functions.

When moving from any operating system to another in SAS, it is imperative to reference SAS Technical report P-195 Transporting SAS files between Host Systems. This will be a great source of information when moving catalogs. Some entry types will not make the move. Code that created a DOS permanent format library will have to be re run. All KEY entry definitions will have to be re-created. PROGRAM entries, FSEDIT screens, CBT and HELP entries, and data will transfer smoothly. In fact, for SAS data sets, simple LIBNAME statements and PROC COPY will be all that is required for the transport. SAS 6.06 can seamlessly read data created using the 6.04 engine. When data sets from the 6.04 release are modified by SAS 6.06 they will be transformed to 6.06 data sets.

The first place to look for changes after moving catalogs and data to release 6.06 is in X commands. If any batch files (DOS statement sequences in a text file with the extension ".BAT") are called in X commands from a DOS SAS application, it is highly recommended that these files are changed to OS/2 command files. This can usually be accomplished in two simple steps. The first step is to rename the file with the ".CMD" extension associated with OS/2 command files. The second step is to change the code calling the batch files

so that it calls the newly renamed command files instead. Most DOS syntax used in the batch files will also be valid in OS/2 command files.

When using PROC CPORT in SAS 6.04 as illustrated in Technical Report P-195, there may be errors. If a catalog is corrupted when creating a transport file in PC SAS, try the "NOCOMPRESS" option on the PROC CPORT.

Under release 6.06 AF and FSP are much improved. Though the transported 6.04 SAS catalogs can be used under SAS 6.06, some modifications to take advantage of the more robust AF and FSP functionality should be considered. In the DOS application all input was required using the keyboard. In the GUI based 6.06, the mouse can be used. An application in which the user never takes his hand from the mouse short of data entry is generally a very easy to use application. When creating an application for end users, the easier to use the better! Selections can be made by using push buttons, icons, or menus.

There are some very useful tools available under the OS/2 release which are not in the DOS release. SCL has grown into a very functional programming language in SAS 6.06. Many functions and options are available [especially] for data sets under 6.06 which were not available under 6.04. These include the WHERE= subsetting, the COPY function, and many more. This is evident if the SAS Screen Control Language Reference, Version 6 Edition and the "Screen Control Language Reference" section in the SAS/AF User's Guide Release 6.03 Edition are compared.

The ATTR selections for screen fields will reveal some new settings for 6.04 users. The most important of the new attributes is the "CHOICE GROUP" attribute. Many screen variable functions can be accessed by creating "Choice Groups". Boxes can be displayed to the user, which he/she simply "clicks" on to activate action fields (which can be instantly detected in SCL using the ISACTIVE function in the MAIN section). Radio buttons can be utilized which only let the user choose one of many possible selections. Push buttons and/or icons can be activated by the user and detected in SCL. An icon can be displayed for a field using the initial field in the ATTR screen. Use two back slashes and a valid number (such as "\112\") from the Appendix in SAS Technical Report P-216.

Perhaps the most important new tool for a user interface to a SAS AF or FSP application is the use of customized PMENU entries. PROC PMENU was introduced in SAS 6.06. It is easy and useful to access PMENU entries in AF and FSP applications. The command line can be turned off in the GATTR section

of an AF window and a PMENU can be substituted. This is accomplished by using the GATTR window "PMENU" setting. The PMENU entry created using PROC PMENU must be in the default AF catalog. Simply type the PMENU name in the field and the PMENU will be displayed with the AF entry. The type of PMENU seen by the user is available in the GATTR section as well, under "WINDOW TYPE". In FSEEDIT entries, the SCL "CALL SETPMENU" function can be used in the FSEINIT section. Any valid four level name can be entered.

PROC PMENU can be a very useful application tool but creating the PMENUS is somewhat of a challenge. To learn the basics of PMENU, refer to the article in SAS Communications First Quarter 1992 and the "Associating Menus with Interactive Applications" article by Yvonne Selby. After this start, try experimenting using other references and the SAS Help system.

Many other advantages of OS/2 will become evident to anyone making the change from DOS. Among these are: The ability to run the DOS version of SAS 6.04 in a window (SAS 6.04 runs a little slower in OS/2's DOS, but is less prone to crashes). The ability to run Microsoft Windows 3.0 (and in the future 3.1) compatible programs is another useful tool. Many powerful Windows based programs are available to OS/2 2.0 users. This includes SAS for Windows. On a 486 system with 16 megabytes of RAM I have run SAS 6.06 for OS/2, 6.04 for DOS, and 6.08 for MS Windows (Developer's release) seamlessly on the same desktop. It is important to note that while the systems did not run as fast as they would have if run exclusively in their native environments, they all executed smoothly and with very little speed degradation. This paper was written using a popular Windows based word processor seamlessly on an OS/2 desktop.

Many other surprises await anyone making the switch from DOS and Windows to OS/2. The power and flexibility of this operating system are comparable to those of some UNIX platforms, and it will run the DOS and Windows software you already own. Often OS/2's DOS compatibility is such that DOS applications run faster and better under OS/2 than they ever could in DOS! Windows applications run just as well as they did in Windows 3.0. Windows 3.1 compatibility has been promised by IBM, and it's sure to be just as usable.

OS/2 or Windows. Why OS/2? For the PC SAS user looking to upgrade his or her desktop, both environments have significant hardware requirements in order to be used as SAS platforms. At least eight

megabytes of RAM and a 200 megabyte hard drive will be in order for either SAS version. There are many advantages to OS/2 over the Windows "Operating System". OS/2 is a complete operating system featuring full 32 bit architecture. Windows is an operating environment which runs under DOS, a very limited 16 bit OS. OS/2 has a faster and more efficient file system. OS/2 is much more crash resistant. Any Windows user can get used to pressing "Control-Alt-Delete" on a regular basis. OS/2 uses preemptive multitasking, meaning more than one process can use the CPU at one time. Windows features cooperative multitasking, which means that one process can use the CPU at once, and multiple applications must switch in a prioritized timing scheme. OS/2 comes with "Workplace Shell", an easy to use object oriented "Mac-like" GUI which is very flexible. Window presents an inflexible "Program Manager" in which groups (or folders) can not be nested, minimized running programs *always* appear at the bottom of the screen, objects can not be sorted, etc. In OS/2, the file system and on-screen program and data objects are closely linked. The Windows "File Manager" completely separates the file system from the program and data icons which the user accesses.

For SAS developers running 6.04 of the SAS system, both OS/2 and Windows represent major technical advances. Both interfaces are graphical and multitask, and both are much better platforms for the SAS system than DOS. Both offer powerful tools and ease of use. The SAS Institute only has a production product for one of the two environments at the time of this writing. This makes OS/2 really the only available choice at this time. More desktops are running Windows, though. This fact alone is a major advantage for the Windows SAS product. There is a much larger installed base of Windows 3.X than OS/2 2.0, though many Windows machines do not have the system requirements which SAS requires. I have used SAS 6.06 for OS/2 and SAS 6.08 Developer's release for Windows. The multithreading, HPFS speed, and preemptive multitasking make OS/2 a faster platform than Windows is while running under DOS. Therefore the SAS version is unsurprisingly faster than the Windows offering. It is important to note that because the Windows product is a beta release, better speed can be expected in production. This still can not allow for the much more technologically advanced OS/2. The plot may thicken when Microsoft introduces Windows/NT and SAS completes a product for that operating system. For now, though, the fastest, most flexible and powerful of the two environments, and therefore the best platform

for the SAS system on an Intel based computer, is IBM's OS/2 2.0.