SOFTWARE CONVERSIONS (WITHOUT ULCERS!)
By Judith H. Mopsik, ARC

INTRODUCTION

The challenge of converting current SAS** production applications to new hardware and software platforms, as well as to the new version 6 release of the SAS system, is now upon us. While some SAS production systems have already been converted from version 5 to version 6 on the PC, the majority of SAS users are still running mainframe and minicomputer applications under version 5. This paper will present approaches for successfully meeting the conversion challenge.

OVERVIEW

The most common conversion situation will be to move current applications from version 5 to the new version 6 of the SAS system on the same computer, under the same operating system. While this appears to be the easiest transition to make, three important questions still need to be addressed:

- How do you approach the conversion in an orderly fashion?
- What are the immediate problems?
- How do you know that your programs still work correctly (without developing a robust test plan)?

SAS programmers have faced a slightly different conversion problem (challenge) since 1984 when the SAS system was introduced on the minicomputer platforms. Applications were ported from one operating system to another; such as an IBM/MVS application that was converted to run as a VAX/VMS application. No, only were the capabilities of the two operating environments different (IBM systems supported the Macro language, but the minicomputer environment did not) but the verification and validation of the resulting application became more complicated when it was discovered that some procedures worked slightly differently: that the ASCII sort sequence, for example, was different than the EBCDIC sort sequence in the IBM mainframe environment.

The third type of conversion concerns the development of an application using PC/SAS and then porting of application to a mainframe or minicomputer environment for production runs. This approach has been demonstrated to be very useful since development costs and time are considerably less on a dedicated microcomputer than on a shared, mainframe system. Since version 6 has many new features that the current mainframe version 5 does not support, a program that has been fully tested in PC/SAS will not necessarily run successfully on the mainframe.

A fourth type of conversion takes place when a production system is moved to a new machine. This has always been a challenge to computer programmers. While the version of the SAS system may not change, the operating system procedures, system editor, and system naming conventions do change, and this results in a major conversion effort that may be difficult to estimate in terms of resources required to complete the job on time and within budget. Such an "opportunity" was presented to the ARC staff on two occasions this past year, when two production systems, developed for two different clients had to be moved from one vendor's mainframe to another vendor's mainframe. Good planning and on-site support helped to guarantee the success of these conversions.

A discussion of the kinds of problems you can expect to encounter will be discussed below. An "outline for success" in meeting the conversion challenge is presented in the discussion that follows. This paper is divided into four sections:

- Plan Ahead: A suggested approach for any conversion.
- Avoid the obvious obstacles
- Getting Down to the Code - Conversion examples
- Conclusions

678
PLAN AHEAD: A SUGGESTED APPROACH FOR ANY CONVERSION

Thoughtful planning of the conversion task before you plunge into modifying source code, will help keep the work organized, and reduce redundant operations. A meeting with the end users of the system will help you identify exactly what the system must deliver. This is especially important when the programming team that will modify the code is not the programming team that implemented the original system. The planning meeting should address the following issues:

- Get a complete understanding of everything the old system did.
- Locate as much documentation as possible.
- Locate all of the source code.
- Benchmark the current production and store all output for later testing.
- Back up the current system and then copy all data and program files, renaming them according to your agreed upon naming conventions.

Backing up the system, and creating a new copy of the source code and data files in a test library, seems like an obvious strategy. It is important to keep the code far away from anyone who might use it by mistake before it is completely tested. Setting up special test libraries will enable a team of programmers to review "converted" code so that they can identify where other changes in their code need to be made.

These basic, administrative tasks can only take place at the start of the conversion project. In a conversion where the hardware and operating system do not change, basic system copy procedures can be used to create a second copy of the source code and data files. In a conversion where you move from one platform to another, the method of transfer could be as simple as a system-to-system download using a communications package such as KERMIT. However, if you are moving between a mainframe and a minicomputer platform, you will have to create TRANSPORT datasets in order to move your SAS files. This capability is provided for in all current releases of the SAS system, and is well documented in the User's Guide. In a conversion where you are moving to a new vendor, but the system remains the same, you will have to copy all code and data to tape, and physically arrange for the transfer. Backup copies of the tapes will be needed, as well. An occasional tape may get left behind in the move!

AVOID THE OBVIOUS OBSTACLES

Once the planning session is completed, and the administrative component of the plan is complete, you can begin to focus on the three major obstacles to a successful conversion.

- The originators of the code, are not available to walkthrough the programs.
- The documentation is not available or non existent.
- The current system "dies" before parallel testing of the two systems can take place.

In order to complete a conversion successfully, one must be ready to hurdle any or all of the above obstacles. For example, once a system goes into production, the original program designers usually move on to new assignments leaving the maintenance tasks to a small team of new programmers. A walkthrough might only take place after the conversion staff have thoroughly analyzed the code and developed some program specifications and data flow diagrams.

As minor, routine maintenance takes place, the documentation, if it existed at all, becomes obsolete. There is always time to change the code, but never time or money enough to update the documentation. Good /*comments*/, but a complete document that included full user requirements as well as a functional specification and program design specifications makes any conversion a lot easier.

Finally, parallel testing of the two systems is vital, otherwise, every single program that was converted will have to go through extensive unit, integration and system testing prior to
running full production again. While some of this testing will go on as the code is changed and comparisons will be made with the benchmark, robust testing of all of the logic is also not something that is usually budgeted for in a conversion. After all, you are only changing a few lines of code to get rid of those "error" and warning messages.

Unfortunately, many computer operations leave only a small window of time open for parallel production activities. It is not uncommon, for example, for both a new and old version of the SAS system to be available for only a short period of time before the new version becomes the production version, and the old version "dies".

GETTING DOWN TO THE CODE

Before you sharpen your pencils or warm up the terminal keyboard (plunge into the conversion) some "global" issues should be addressed. Modifying a lot of code without taking the appropriate precautions will require more time and resources in the long run.

- Run a several test cases through both systems and assess the differences. The opportunity to have both the old and new systems concurrent may not last long.
- Commit yourself to minimizing changes and avoiding total rewrites of the programs.
- Identify common code (or "tool boxes") in the system and convert it first.
- Prototype one complete "path" through the system.

The first step, of course, is to simply run the programs under the new version (changing the JCL or other system commands so that you can at least get the code to compile), setting OBS=0. This run will identify what will and will not work, and what will generate the infamous WARNING messages. Once you have identified which features are incompatible, you will be able to evaluate how much code really needs to be "fixed". Once you have taken a "global" look at the environment, you will want to make a list of the known differences between the two systems. You will probably discover more as you get involved with modifying the code.

Minimizing changes to the code, and avoiding the temptation to rewrite large sections of new code is very important. In tackling the conversion problem, you are only trying to keep the system running. Any significant changes will in all likelihood impact other modules in the system, and leave you with a big "mess". If your goal is to update the system and incorporate the latest features of the newest version of the SAS system, then a new design might really be in order.

By identifying common code, you can minimize the volume of work to be done. If the original code is repetitive, a few user-defined Macros may collapse several identical datasteps into one that could be implemented as a Macro, and only require only a single set of changes.

Prototyping the system involves converting one complete path of code and getting it to execute correctly. This would be analogous to the integration testing that was done during the original development. This will help you to identify any major problems that the conversion will bring. Earlier this year, ARC staff were asked to convert a major survey system from version 82.4 to version 5.16. Our client's computer center was finally putting version 5 into production! Knowing that quotation marks had to be present for all TITLE, LABEL and FORMAT statements, we quickly reviewed the scope of this activity and slated it as our first task. What seemed like an obvious solution (putting quote marks appropriately on all such strings of data) turned out to be complicated, and several attempts were made just to determine the best (i.e. most solid, expedient) solution. The original code, first written under version 79.6 used "old style Macros" to accomplish substitutions in TITLE, %INCLUDE and FORMAT statements:

MACKO Method '3A': %

TITLE This is for Method;

The first change needed was to put 'Quotation marks' around the TITLE string to eliminate the 620 WARNING NOTE in the SAS Log. However, the following statement

TITLE This is for Method Method;
did not resolve the Macro variable because the SAS system considers all text within single quotes in a TITLE statement to be used "as is". Furthermore, using the DQUOTE option, the SAS compiler did not identify METHOD as a macro variable, since the Macro processor requires a % or & token. Several attempts at correcting this led to keeping the original unquoted TITLE, and ignoring the 620 WARNING.

While it was suggested that we simply convert to "new style Macros", this was not really feasible, since more than 50 similar types of Macros like «Method» were used throughout the system (60,000 lines of SAS code) and a major overhaul of the code would have been required. Complex statistical routines and a user-written procedure were involved as well. Converting to the new Macro language was not a viable solution. Subsequent testing would have been exhausting.

Furthermore, there were only six weeks to complete the task before the computer center removed Version 82.4. Consequently, any solutions requiring new Macros was not feasible for this application. The conversion to the new Macro system will wait until Version 6.

Prototyping will also enable you to gauge how long the entire task will take. A plan for dividing up the work, or "farming it out" becomes possible, once you know exactly what has to be done and how it has to be implemented.

EXAMPLES

1. SAS version 5.18(Mainframe) => SAS version 6.03 (PC)

Environment: An old mainframe application, first developed under version 79.6 was to be ported to a PC

Problems: Old style Macros cannot be used. User-written procedures developed before version 5 must be re-written.

All LABEL, TITLE, and FORMAT statements require appropriate quotation marks

COMMENTS change

Solution: Read the Technical Reports very carefully!

2. SAS version 6.03 on the PC => SAS version 5.18 on a mainframe

Environment: A mainframe application was to be developed on a PC and then ported to the mainframe, in another city 250 miles away. The system was an end-user reporting system with menus for report selection and parameter selection (e.g. variables for sorting, printing, aggregating).

Problems: SAS/AF and SAS/FSP make use of the Screen Control Language (SCL) in version 6. Even if you do not use SCL in developing the code, the SAS supervisor does.

Solutions: In order to port your programs successfully you will have to create raw files of your screens, "INCLUDE" the file as text under AF or FSP, and then edit the screens in MODIFY mode under 5.18.

3. SAS version 5.18 on the IBM mainframe platform => SAS version 5.18 on the VAX/VMS platform

Environment: CLISTS were used to produce SAS reports for batch submission. Under the VAX system, ARC proposed the use of SAS/AF for a similar batch reporting system.

Problems: ISPF and CLIST features would have to be simulated so that users could select reports to be submitted, along with variable selection, sort order and aggregations.

ARRAY references were used extensively.

Solutions: SAS/AF was used to simulate the CLIST design, and %LET, SYMPUT and SYMGET were used to create Macro variables. SAS/AF code generated the DCL command files that would submit batch jobs.

Implicit ARRAYS had to be converted to explicit ARRAYS, and all DO OVER code had to changed. The DIM function substituted for a DO OVER when the ARRAY dimension was unknown.

All JCL references had to be changed and LIBNAME statements added to reference SAS datasets. LIBSEARCH statements had to be added to reference user-created formats stored in other directories.
4. Porting the same system from one vendor to another

**Environment:** A production on-line database retrieval system, consisting of hundreds of tape datasets as well as massive on-line datasets. The system to be moved was in production and could not afford to be "down" for many days.

**Problems:** New vendor required different dataset naming conventions so all JCL would have to be changed.

Many components of the system dynamically wrote system commands (i.e. JCL) that would also have to be changed.

SAS options that were standard at the old vendor were not the standard set at the new vendor.

Old vendor had WYLBUR for the system editor, the new vendor used ISPF.

Costing formulas were very different, and the system was "tuned" for the old vendor's cost accounting formulas.

**Solutions:** The new vendor converted all obvious JCL statements to their standard naming conventions. Only dynamically generated JCL had to be converted manually.

The new vendor provided immediate training classes for anyone new to ISPF, along with complete documentation.

All programming was frozen for a period of two weeks so that the system could be brought back into production and completely benchmarked.

The problems of "tuning" the system to the new cost formulas took time, because most of the consequences were only realized after the first bills started to arrive. In this particular case, large batch jobs, run at a very low priority were one-tenth the cost that they had been under the old vendor. However, interactive costs (against the on-line database) were significantly higher, and the staff worked to modify the new system environment parameters to reduce these costs. This took place after the system was back in operation.

**CONCLUSIONS**

There are many advantages to making a conversion to a new system. First, there is the opportunity to "clean up" old code. There are always parts of a system that did not get the attention that they deserved during the original life cycle development. The conversion offers an excellent excuse to correct the problems. Second, the conversion presents an opportunity to "modernize" old code, using newer techniques and procedures to achieve the same results. While this is more viable in a small system, you may discover great efficiencies by using a different reporting procedure, for example, or taking advantage of the new reporting features of PROC SUMMARY or PROC TABULATE for example. Third, you can clean up your source code libraries, getting rid of redundant code, or ad hoc code that was just "hanging around" but only used once. If you are not going to convert it, throw it away.

The conversion also opens up new opportunities for the programming staff. This could be the time to get end users to agree to some menuing approaches for running production jobs and producing reports. This gives the users access to the system, but keeps them out of the code!

The transition to a new system will be manageable if:

- You separate the Conversion work from ongoing production activities
- You plan to systematically change code, testing as you go along
- You run parallel production systems to insure the quality of the new code

The transition will be cost efficient if you

- Prototype first to assess the scope of the job
- Document as you go along so that you do not make the same mistakes too many times over
- Take advantage of new features that reduce I/O time, or incorporate report output into the PROCEDURES

If you can easily control the transition and stay within budget, your conversion will be a success.

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