

Conversion to SAS from Other 4GL Languages

Rick Nicola SPS Software Services Inc

Abstract: The introduction of PROC REPORT, as well as the SAS/ACCESS® products, has made SAS a viable alternative to many other report writers and products tagged as "fourth generation languages." Those shops who wish to standardize on a single product to save software costs and direct their employee skills, will find SAS the logical choice. The full displacement of software products is, however, easier said than done. This paper will present a generalized method for converting from such products to SAS. Specific methods used for EASYTRIEVE (batch) conversion and the conversion from FOCUS will be presented. Pitfalls, problems, shortcomings, and circumventions will also be reviewed.

Many data processing installations that have a broad user base, have made an effort to provide their non-DP clients with "user-friendly" tools. Unfortunately, as time has passed, many of these tools have not kept pace with the times and prove costly on several fronts. This paper presents SAS as a solution to this dilemma. We look at why we might wish to convert from other products to SAS and examine conversion steps and service aids that can expedite this process.

Why Convert: We can sight three primary reasons for using a single product to provide "user friendly" computing. First, of course, is cost. Second is to focus skill sets and training. A third reason for conversion deals with strategic issues. While the first two reasons might suggest concentrating usage on one of several existing products, SAS alone meets the strategic criteria.

When we talk about the cost of software, we must consider costs far beyond the annual renewal fee. In fact, in trying to compare the cost of converting against the cost of continued use of multiple products, one might consider the following comparison:

Cost of Conversion	V.
Software annual renewal fees	
+ Continued software support	
+ Cost of redundant training	
+ Cost of redundant data / access.	

The cost of renewal is perhaps the only expense that is immediately evident. Including the costs of ongoing support and redundant user training may seem a bit of a stretch, since the same people may be involved in supporting multiple products. However, supporting actual product usage and providing in-house or external training does represent actual costs. Data access costs may not be a factor in all cases. If, however, the same data must be "touched" on a regular basis by more than one product, one would expect that using a single product to meet all needs would be more cost effective. The value of each cost element must be determined for individual situations.

The key point in this comparison, however, is that the conversion costs are "one time", while not converting represents recurring costs. Therefore, there will always be a payback. The question is whether this payback will be over a reasonable period of time.

The consideration of skills requirements in a multiple product environment is certainly a part of training costs (and is included above.) There are other elements of skills development that should be considered that are not directly related to costs or have cost parameters that are not easily measurable. While we are always concerned about products being user friendly, it is apparent that the more one uses a product, the greater his skills become. Just as it is reasonable to expect skills to grow with continued usage, one would expect skills to grow more rapidly when usage is focused in a specific area or on a specific package. It is expected that this "tighter" skill set would yield more rapid and more elegant solutions as those skills grow.

The final element in examining why one would consider focusing usage on SAS deals with strategic issues. This area points almost exclusively to using SAS as the single product. The strategic issues really fall into three categories:

- | |
|-------------------------|
| 1. Platform crossover. |
| 2. Legacy systems. |
| 3. Long term viability. |

Because of the recent upheaval in data processing, many companies are reluctant to address any new needs. There seems to be an underlying expectation that new systems, as well as established base systems, will need to be rewritten, replaced or "re-engineered". Thus, the use of the data by non-DP user groups or "open shop" applications would be made obsolete by the process of base system replacement. Furthermore, "re-engineering" often seems to imply a platform change, specifically to some version of a client / server environment. In any case, SAS provides truly unique facilities for running applications on multiple platforms. In fact, connectivity products allow for storing data on one platform and presenting it on another. Most importantly, however, SAS provides entry into numerous databases. No matter what the physical (hardware / software) or data (database, flat file) configuration, a SAS user inquiry or program can be easily adapted.

This adaptation can be applied not only to newly developed systems, but also to those "legacy" systems that have been in place for some time. While files and data may be reformatted or "re-engineered", the base measures remain essentially the same. Thus, legacy SAS programs may require changes only to the initial access of "re-engineered" files, and this change in access may be made considerably less difficult by the various access tools.

Perhaps the most important reasons for selecting SAS has to do with long term viability. It seems that this issue has been frequently ignored in recent years. We are often so taken by the apparent ease of use of the latest PC product or access tool, that we often fail to consider the long term viability of the vendor. This is disturbing, particularly when you consider that some of these vendors market very few products and are wholly dependent on new sales of the next release of their existing products to generate revenue. Several such companies are on the brink of "going under" and the viability of many others is certainly in doubt.

SAS, on the other hand has been in existence for nearly twenty years. If we were to try to characterize the growth in changes in SAS, we might consider the following logic. Twenty years ago, SAS was a good data handler, but lagged the "report writer" software of the day in ease of use and flexibility. Ten years ago, many of these report writers failed to

adapt and fell into disuse, while SAS improved on its data handling techniques. During this period a decade ago, some products began to offer multiple platform products (PC and mainframe). Again, SAS lagged in this technology. However, today, ten years later, SAS products have far surpassed the "players" of 1985. And, while today's SAS products seem to slightly lag the front runners in "object oriented" facilities, I would expect that in 2005 it will continue to expand and grow while some of today's "players" have diapered.

Stated quite simply SAS is the foremost strategic tool in computing today. The continued, renewal based revenue structure and track record for meeting changes in data processing has distinguished it as the software that will continue to meet users needs.

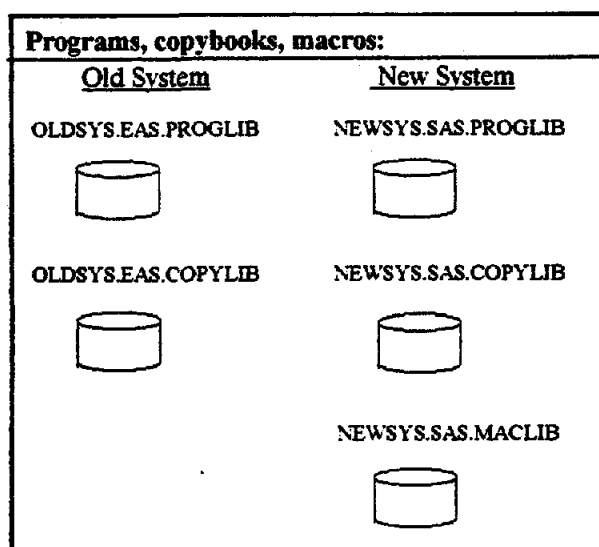
How to convert: In looking at how to convert to SAS, we discuss a preparatory step and give two examples or types of conversion. The preparatory step deals with identifying programs which need to be converted and how the "new system" might be laid out for ease of conversion, use, and maintenance. The conversion examples sighted are a batch "EASYTRIEVE" run and a FOCUS "table talk" system.

Perhaps, the most important part of doing such a conversion may be the "front end". Before starting, the analyst must first identify which programs are actually active. The cloning of similar programs, particularly in interpretive languages, tends to lead to numerous similar versions of the same program. If this is true, eliminating the unused programs will reduce the conversion effort considerably. In addition to which programs are used, the analyst performing the conversion must determine when they are run, how they are run, and who sees the results.

Once the "used" programs have been identified, one can then identify the data that is used. The analyst must consider how the data was used in the old system and how it might be best used in the new system. Should the new application read in "raw data" each time? Should one use a "view" into an existing database? Should a SAS based "warehouse" of data be built and used? Considerations in this determination include ease of access, performance and the duplication of data.

The most difficult preparatory step is in determining whether to mimic the old system, or attempt to improve it. In dealing with systems put together by casual or departmental users one often finds some undesirable characteristics. First, many programs or sequences of programs have been built and changed over time by a number of individuals. Second, some programs require regular maintenance (to change date or other information). Finally, many systems may have programs that contain similar or identical code because this code has been "cloned" from program to program. When converting such programs to SAS, there is a great temptation to use SAS facilities to externalize redundant code, use date functions to avoid constant changes, and to make other coding improvements. If one chooses to do this at conversion time he will greatly increase the conversion manpower requirements. In addition, it will make validation against the "old system." more difficult.

Even if the analyst chooses to make few changes, he might still consider setting up a program structure that at least allows for improvements in the future. For example, suppose the old system uses a report generator with copybooks. Perhaps the new system should have the structure for programs, copy books, and macros. Consider the old and new "MVS" datasets below.



Type 1: Batch Conversion (Report Writers).

The first actual example of converting the package code to SAS, deals with an EASYTRIEVE batch program. In order to "convert" such a program, we are concerned about how we deal with the data and how we can most easily convert the program itself.

In converting the "read in" portion of the data, we can use a SAS service aid called "COB2SAS." It is assumed that, if there is an EASYTRIEVE program, there are COBOL record definitions. The COB2SAS program reads COBOL "FD's" and creates equivalent SAS input statements. In actuality, they are not truly equivalent. If the COBOL names are too long, COB2SAS uses a pretty good name compression mechanism (that builds reasonably sounding eight character names.) the output of the "conversion" is a starting point. The user should then change the input routine to make use of the power of SAS date handling and other facilities. Finally, decisions have to be made on how to handle fields spanning more than 200 characters, redefined fields, and other tasks.

```
//CONVERT EXEC SAS
//COB2SAS DD DISP=SHR,DSN=COB2SAS.PROGRAMS
//INCOBOL DD DISP=SHR,DSN=COBOL.COPYBOOK(MEMBER)
//OUTSAS DD DISP=MOD,DSN=SAS.INPUTS(MEMBER)
//SYSIN DD DISP=SHR,DSN=COB2SAS.PROGRAMS(R2MVS)
```

Where

INCOBOL DD contains the original COBOL FD's.
OUTSAS contains "first cut" SAS Input Statements.

In working with COB2SAS, the user will learn to make adaptations to the starting code by handling dates properly and eliminating or changing variable redefinition. Once the user is satisfied with the input routine, a decision must be made on how to store the data. Should the user "read in" the raw data each time, as is done with the report writer or is the data used frequently enough to justify warehousing it in SAS format. There is a middle ground that may be appropriate, particularly if the data is stored in another database format. One might consider building a "VIEW" into the data using one of the SAS / ACCESS[®] tools or even using a data step view for "flat" files.

The sample below reads in raw data. The EASYTRIEVE code is shown on the left, with the SAS equivalent code using PROC REPORT on the right. Without going into too much detail, note that PROC REPORT has many of the same characteristics as the report writer.

Type 1: Batch Conversion

Program conversion: EASYTRIEVE:

```

FILE CSHKREPT
CSHKREPT-REC          01 80 A
CODE 01 02 A HEADING ('REC.' 'CODE')
LINE-NO 03 02 A HEADING ('LINE' 'NO.')
STORE 05 05 A HEADING ('STORE' 'NUM')
CHAR-R 10 01 A HEADING ('CHAR' 'R')
DATE 11 06 A HEADING ('WEEK' 'END')
DESC 17 21 A HEADING ('DESCRIPTION')
ACCT 47 05 A HEADING ('ACCT' 'NUM')
AMNT 53 09 N 02 HEADING ('AMOUNT')
WEEK 69 03 A HEADING ('WK#')
*
DEBITS      W 10 N 2 MASK (BWZ 'ZZ,ZZZ,ZZ9.99')
              HEADING ('DEBITS')
CREDITS     W 10 N 2 MASK (BWZ 'ZZ,ZZZ,ZZ9.99CR')
              HEADING ('CREDITS')
DIFFERENCE W 10 N 2 MASK (BWZ 'ZZ,ZZZ,ZZ9.99-')
              HEADING ('DIFF') *

JOB INPUT CSHKREPT
CREDITS = 0
DEBITS  = 0
IF ACCT NQ ' '
  IF AMNT GE 0
    DEBITS = AMNT
    CREDITS = 0
  ELSE
    CREDITS = AMNT
    DEBITS = 0
  END-IF
END-IF
IF ACCT NQ ' '
  DIFFERENCE = AMNT
ELSE
  DIFFERENCE = 0
  END-IF

PRINT REPORT1
GOTO JOB
*
REPORT REPORT1 PAGESIZE 55 SPACE 1
CONTROL STORE WEEK
TITLE 1 ' RUN TIME ' SYSTIME +
CASH REPORT STORE DETAIL LISTING '
LINE 1 STORE DATE WEEK DIFFERENCE LINE-NO ACCT
      AMNT DEBITS + CREDITS
*
BEFORE-BREAK. PROC
*
  IF DIFFERENCE = 0
    DISPLAY '***IN BALANCE *****'
  ELSE
    DISPLAY '***NOT IN BALANCE *****'
  END-IF
END-PROC
*
AFTER-BREAK. PROC
  DISPLAY
*****
END-PROC
*
```

Program conversion: SAS:

```

PROC FORMAT; /* SPECIAL FORMATS.DONT PRINT 0'S */
PICTURE CREDITS OTHER = '0000,000,000.00CR';
PICTURE DIFFER LOW-<0 = '0000,000,000.00-';
              0-HIGH = '0000,000,000.00+';
PICTURE DEBITS OTHER = '0000,000,000.00' ;

DATA CSHKREPT ; /* START DATA STEP */
  INFILE CSHKREPT ;
  INPUT @03 LINE NO 2.
        @05 STORE $5.
        @11 DATE $6.
        @17 DESC $21.
        @47 ACCT $5.
        @53 AMNT 2D9.2
        @69 WEEK $3.;
  XAMNT = AMNT ;
  XACCT = ACCT ;

PROC SORT ;
  BY STORE WEEK LINE_NO ;

PROC REPORT DATA= CSHKREPT HEADSKIP NOWINDOWS;
  TITLE1 "CASH REPORT STORE DETAIL LISTING";
  COLUMN STORE WEEK DATE ACCT AMNT DIFFER
        LINE_NO XACCT DESC XAMNT DEBITS CREDITS;
  DEFINE STORE/ORDER FORMAT=$5. WIDTH=5
        'STORE/ NUM' ;
  DEFINE WEEK/ORDER FORMAT=$3. WIDTH=3 'WK#';
  DEFINE DATE/DISPLAY FORMAT=$6. WIDTH=6 'WEEK/END';
  DEFINE LINE_NO DISPLAY FORMAT=$2. WIDTH=6 'LINE/NO';
  DEFINE XACCT/DISPLAY FORMAT=$5. WIDTH=5 'ACCT/NUM';
  DEFINE ACCT/DISPLAY NOPRINT ;
  DEFINE DESC/DISPLAY FORMAT=$21. WIDTH=21
        'DESCRIPTION' ;
  DEFINE XAMNT/ANALYSIS SUM FORMAT=DIFFER. WIDTH=17
        'AMOUNT';
  DEFINE AMNT / ANALYSIS SUM NOPRINT;
  DEFINE DIFFER/COMPUTED FORMAT=DIFFER. WIDTH=15
        'DIFF';
  DEFINE DEBITS / COMPUTED FORMAT=DEBITS. WIDTH=15
        'DEBITS';
  DEFINE CREDITS/ COMPUTED FORMAT=CREDITS.WIDTH=15
        'CREDITS';

  COMPUTE DEBITS ;
    IF ACCT ^= ' ' AND AMNT.SUM GE 0
      THEN DEBITS=AMNT.SUM; ELSE DEBITS = 0 ;
  ENDCOMP ;

  COMPUTE CREDITS ;
    IF ACCT ^= ' ' AND AMNT.SUM GE 0
      THEN CREDITS = 0;
    ELSE CREDITS = AMNT.SUM;
  ENDCOMP ;

  COMPUTE DIFFER;
    IF ACCT ^= ' ' THEN DIFFER = AMNT.SUM ;
    ELSE DIFFER = 0 ;
  ENDCOMP ;

  COMPUTE AFTER WEEK ;
    IF -0.00000009 < DIFFER < 0.00000009 THEN
      DESC = '***IN BALANCE *****';
    ELSE DESC = '***NOT IN BALANCE *****';
    LINE @2 DESC $21. ;
    LINE @2 41'***'; LINE @2 41'***';
    LINE @2 ' ' ; LINE @2 ' ' ;
  ENDCOMP ;

  BREAK AFTER WEEK / OL SUMMARIZE ;
  BREAK AFTER STORE / OL SUMMARIZE ;
  RSBREAK AFTER / OL UL SUMMARIZE;
```

SAS Report...(EASYTRIEVE version is are very similar)

CASH REPORT STORE DETAIL LISTING									
STORE WEEK	DATE		LINE	ACCT					
NUM	END	DIFF	NO	NUM	DESCRIPTION	AMOUNT	DEBITS	CREDITS	
99797	702	071488	01		CLOSING READING	687,341.16+			
		071488	02		OPENING READING	456,704.99+			
		071488	03		RUNG UP XXXXXXX	230,636.17+			
		071488	05		MERCHANDISE RET	704.11+			
		071488		39 36501	PHARMACY DEPART	7,482.98-		7,482.98CR	
		071488		40 36701	VIDEO DEPARTMEN	2,751.90-		2,751.90CR	
		071488		80 71931	CASH OVER	17.08-		17.08CR	
-----						-----			
99797	702				**IN BALANCE **	2,840,996.98+			
-----						-----			
99797						2,840,996.98+			

Type 2: Code Generating Systems.

FOCUS is a fairly powerful package that allows for creating custom reports using a facility called "Table Talk." Table Talk builds FOCUS code that actually does the retrieval / reporting. This concept is very similar to the SAS Query Window or some of the other code generation facilities in SAS / ASSIST®.

SAS lagged FOCUS in its availability on the PC and in a few other areas. However, it seems as though new SAS facilities offer greater promise for the future and it is therefore the tool of choice.

The process for supplying a "interactive build" consists of two steps: providing access to the data and providing the build process.

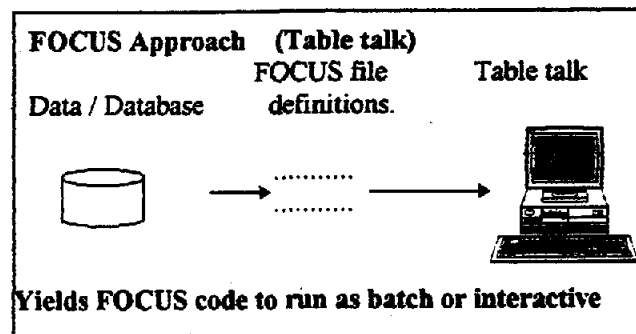
Data access will vary based on the size and type of file (i.e. flat file, relational database, etc.) Since this is an interactive process, there is greater concern for performance as measured by response time. For flat files, COB2SAS may again be used as a starting point. For databases, SAS / ACCESS® may be used for direct access or extraction into a SAS data set.

Data access options reduce to three basic choices:

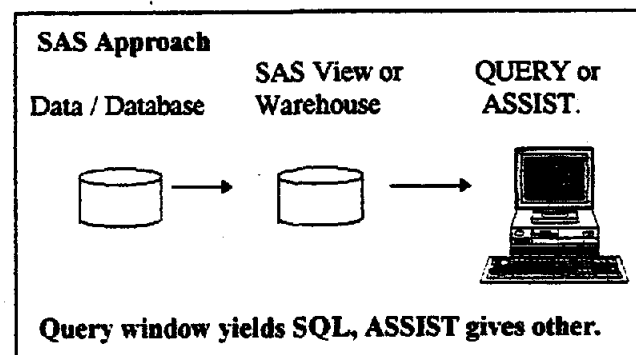
1. Build a data step view into the file (flat files)
2. Build an access view into a database (assumes SAS / ACCESS® available)
3. Maintain the data in a data warehouse (in SAS format)

In all cases, it is conceivable that an SQL View or an SQL Joined View could be built. However, in the case of using views into flat files, response time might prove prohibitive.

The figure below shows how FOCUS accesses raw data using table talk:



SAS provides a similar flow, however, the VIEW v. warehouse elements provide several options.



Sample Focus Table Talk Screen:

And in the next report column	Select a
No more	SSCNBR
And PRINT the individual values of	SSNBR
And WRITE the PERCENT of	NAME16
And COMPUTE the RATIO of	INITIALS
And COMPUTE the PRODUCT of	LASTNAME
And COMPUTE the DIFFERENCE between	STATUS
Underneath the last column	BARGUNIT
(MORE)	LOCATN

TABLE FILE XEAC214C
PRINT LASTNAME AND LOCATN AND STATUS

Resultant FOCUS Output

PAGE 1		
LASTNAME	LOCATN	STATUS
GILLOTT	638	N
HANDLEY	572	N
PEBA	1	N
JEAN	6	H
BOCKMAN	528	X
CHAMBERLAIN	771	H
DECOTEAU	559	H
OSBORN	95	X

Sample SAS Query Screen:

SQL QUERY WINDOW COLUMNS

File Actions Locals Profile Globals Help

Select column(s) for query:

Available Columns	Selected C
< COUNT(=) >	LASTNAME
* XEAC214C * <al	STATUS
SSNBR	LOCATN
INITIALS	
LASTNAME	
STATUS	
BARGUNIT	
LOCATN	
RETCOD	
COBRAFLAG	

Column Alias/Label

Column Formats

Resultant QUERY (SQL) Output

THE SAS SYSTEM			11:39 TL
LASTNAME	STATUS	LOCATN	
GILLOTT	N	638	
HANDLEY	N	572	
PEBA	N	1	
JEAN	H	6	
BOCKMAN	X	528	
CHAMBERLAIN	H	771	
DECOTEAU	H	559	
OSBORN	X	95	

Summary: SAS presents the opportunity to concentrate DP and user based 4GL systems wholly on SAS. Service aids, such as COB2SAS give a starting point for building SAS "Warehouses" or data step "VIEWS." and SAS/ACCESS® allows for direct entry into many databases. PROC REPORT is a viable alternative to report writers, while the QUERY window and SAS/ASSIST® offer alternative to other products ad-hoc "building" techniques.

However, a conversion such as this should not be taken lightly. While costs can usually be justified, such conversions are not trivial. Proper project planning should precede the effort, with particular

concern for system performance and the user's comfort level with the new SAS system.

References for: Conversion to SAS from Other 4GL Languages

MWSUG October, 1995 Rick Nicola
SPS Software Services Inc. Canton, Ohio

SAS Tech Report P-258- Using the REPORT Procedure in a Nonwindowing Environment, 1993, SAS Institute, Cary, NC

EASYTRIEVE PLUS, Student Guide, 1988, Pansophic Systems, Inc., Oak Brook, IL.