Paper 211-25

SAS® For COBOL (and Other) Programmers: How To

Doug Zirbel, Modis, Inc., Bloomington, MN

ABSTRACT

For all its power and behind-the-scenes success, SAS software is still unknown to many programmers and I/T management. This paper provides tools you can use to solve this problem in your company or area by offering a short SAS class which gets programmers up and running quickly.

INTRODUCTION

SAS software is widely used in the mainframe Information Technology world. It is the eighth-largest software company in the world. It has won awards.

Yet if you have used SAS software in this environment, you may find yourself having to explain to your colleagues what SAS is. However, by offering a short class – even as "brown bag lunch" sessions – to programmers who aren't familiar with SAS, you can go a long way toward answering that question.

From my experience teaching such classes and from the advice and feedback of attendees and others, I can offer several "tools" you might be able to use in developing a custom SAS course at your site.

They are:
1) FAQs (Frequently-asked-questions) about SAS
2) Myths about SAS
3) Feedback from previous students and others
4) Charts of COBOL / SAS similarities

FAQs

Q: Can I load VSAM files?
A: Use OPTION VSAMLOAD=YES;

Q: Can I do a transaction-master (external) file update, just like in COBOL?
A: Yes, use a Data _null_; program with INFILE, INPUT, FILE, PUT, and IF statements.

Q: Can I do a transaction-master SAS file update, just like in COBOL?
A: Yes, use either MERGE, UPDATE, or MODIFY statements, *or* PROC SQL.

Q: Can I read Roman numerals just like other numbers??
A: No. *But*, you can write them with the ROMANw. format!

Q: Can I get SAS at Best Buy?
A: No. Call your local SAS office.

Q: Can I use SAS along with SyncSort, FileAid, EZTrieve, awk, REXX, CLIST, Lotus Notes, JAVA, ODBC, IEBGENER, C++, VB, DB2, IMS, IDMS, Access, Excel and TCP/IP?
A: Yes. And in some cases *in place of*.

Q: Can I call a subroutine like I can in COBOL?
A: Yes, with CALL MODULE('subroutinename',arg1,arg2…).

Q: Can I do string functions just like everything I can do with INSPECT?
A: SAS offers a wide range of character functions. For example, FLD1=*TRANSLATE(FLD1,TO-CHARACTERS,FROM-CHARACTERS*)

Q: Can I use SAS instead of COBOL for on-line transaction processing (OLTP) with CICS?
A: SAS has no provision for this, and although SAS / AF is screen-based, it is better suited to on-line analytical processing (OLAP).

Q: Can I get a randomly-generated subset of 100 recs from a 2,000,000-rec input file for testing purposes?
A: You can do it with one line of code. To do so in COBOL would require a more complex subroutine or paragraph. Check out the RANUNI function in the SAS Language manual. Example: IF RANUNI(0) LE 0.00005 THEN OUTPUT;

Q: Can I write Windows or Unix graphics-based screens to access my mainframe data?
A: Yes. That's SAS/AF, but out of the scope of an introductory mainframe-based course.

Q: How do I find out exactly what I've got installed at my site?
A: In SAS Data Manager or in batch, run PROC SETINIT;

MYTHS ABOUT SAS

Myth: SAS cannot process an external file until it is read into a SAS dataset.
Truth: "Data _null_; " programs can be as fast and even faster than COBOL for non-SAS external file processing.

Myth: Given a number of sorted flat files to be merged and written as a flat file, SAS will run out of space converting all of them to SAS files first.
Truth: SAS can utilize data views, which take virtually no space.

Myth: Custom-formatted reports cannot be written in SAS because SAS has its own output style.
Truth: This would be to some extent true with certain procs. Again, "Data _null_" steps allow tremendous flexibility, while PROC REPORT is a rapid custom reporting method (e.g. 10 – 20 lines of code), and an even simpler PROC PRINT can be coded in two or three lines!

Myth: SAS is a resource hog.
Truth: By and large, this is a result poorly written programs, the same way as in COBOL or any other language.

Myth: SAS arrays are so different from COBOL tables that no COBOL knowledge can be applied to use them.
Truth: SAS arrays are remarkably similar to COBOL tables.

Myth: SAS – it's that statistical software isn't it? – we wouldn't have any use for it here.
Truth: SAS can do almost anything COBOL can do, with far less time spend coding. SAS's statistical functionality is only one of the dozens of roles it fills.

FEEDBACK FROM STUDENTS AND OTHERS

"When do you use SAS vs. COBOL?" (The popular answer seems to be, "when you want to get something done quickly.")

"SAS has four core technologies:
Multi-Vendor Architecture, which means it can function with many different brands of software, Multi-Engine Architecture, which means it can run on many different operating systems, Rapid Application Development, which means you can sometimes do something in one day in SAS which might take up to three weeks in COBOL, and Web enablement, which COBOL simply doesn't have."

"Show a COBOL program and a SAS program side-by-side."

"Define where SAS fits in the computer languages spectrum – it's a fourth-generation language."

"The examples were good. Most of us take courses, but don't get to use what we learned for a while. Maybe a cheat-sheet."

"It would [be helpful] …to have some lab-time…" (This comment was from an overheads-only course due to a lack of training computer arrangement. It serves to emphasize this need!)

CONCLUSION

Presenting a talk or a class on SAS at your worksite takes preparation and should be adapted to build on the existing knowledge of those who will attend. Make the advantages (and potential disadvantages) of SAS clear. Develop a handout which will be instantly useable long after most of the course has been forgotten.

TRADEMARKS

SAS is a registered trademark of SAS Institute Inc. in the USA and other countries.

REFERENCES

SAS Language: Reference, Version 6, First Edition, #C56076, Cary, NC: SAS Institute Inc.
SAS Companion For The MVS Environment, #C55108, Cary, NC: SAS Institute Inc.

CONTACT INFORMATION

Doug Zirbel
Modis, Inc.
Bloomington, MN
zirbel@computerpro.com
(218) 525-5518

ACKNOWLEDGEMENTS

I am especially grateful to
Michael Sperling at the City of Phoenix,
Phil Weiss at SAS Institute in Phoenix,
many students, and
SAS-L discussion group participants.

APPENDIX

(next pages)

COBOL – SAS COMPARISONS

COBOL	SAS	NOTES
SELECT TRANSFILE ASSIGN TO DD1	INFILE DD1;	Names DDNAME of ext file, but SAS calls it DD1 throughout, also treats it as an OPEN statement at compile time. Closes it at end.
Not available	FILENAME xyz 'YOURTSO.TRANS.FILE' DISP=SHR; FILENAME xyz '/usr1/misc/trans.txt'; FILENAME xyz 'c:\My Documents\trans.txt'	Like a DD statement in JCL, but within SAS code itself … a UNIX version … and a Windows version
READ TRANSFILE INTO WS-TRANS	INPUT @01 field1 $CHAR18. @21 field2 $CHAR6. @28 fld3 $CHAR4.;	Reads external file, but SAS puts the file layout here rather than in a WS area
01 WS-TRANS. 05 FIELD1 PIC X(18). 05 FIELD2 PIC X(6). 05 DATE1 10 YY PIC 99. 10 MM PIC 99. 10 DD PIC 99.	INPUT @01 field1 $CHAR18. @21 field2 $CHAR6. @28 date1 $YYMMDD6.;	Reads external file, but SAS puts the file layout here rather than in a WS area
WRITE MASTERFILE FROM WS-TRANS	FILE MSTRFILE; PUT _INFILE_;	This particular example writes the output file from the whole WS-TRANS file
WRITE MASTERFILE	FILE MSTRFILE; PUT @01 field1 @25 field2 @120 fld3;	This writes 3 fields to an output rec
IF *condition-x* cobol-verb(s) ELSE cobol-verb(s) ENDIF	IF x THEN DO; Sas-verbs END; ELSE DO; Sas-verbs END;	You can have multiple nested-ifs in both COBOL & SAS of course.
PERFORM 2000-PROCESS-INPUT THRU 2000-EXIT.	LINK PROC2000; sas-verbs RETURN; PROC2000: (← that's a colon) sas-verbs RETURN;	Where COBOL thrives on hierarchical structure, SAS prefers fall-through type logic. However, there are definitely times when it's appropriate to do perform-type logic in SAS.
ADD TOT1 TO XTOTAL GIVING YTOTAL.	YTOTAL=TOT1 + XTOTAL;	Similar for multiply, subtract, divide, etc. + - * / **
ADD TOT2 TO GRAND-ACCUM.	GRANDACM+TOT2;	TOT2 varies with each input rec, but GRANDACM keeps getting bigger
WS-A-GROUP OCCURS 10 TIMES.	ARRAY A_GROUP(10) GROUP1-GROUP10;	Much more flexible than can be shown here.
	IF GENDER='M';	Called a subsetting-if in SAS… it means if your input rec's GENDER var ='M' then keep the record for writing to output file, otherwise don't write it to the output file.
STOP RUN.	RUN;	SAS automatically stops processing when it gets to the end of input unless you tell it otherwise. RUN marks the end of the code or module, and also writes out 1 output rec, then returns to the top of the DATA statement to begin with the next input rec.

CALL 'ILBOAN0'	ABORT ABEND 16;	This abends & sets RC=16
MOVE A TO B	B=A;	A stays the same in both COBOL and SAS (doesn't get wiped out).
MOVE 'A' TO NUM-FIELD1.	NUMFLD1='A';	0C7 in COBOL, but just an error message in a SAS log
WHERE (condition)	WHERE (condition)	Can be used on SAS datasets in data steps and procs…doesn't need to be relational DBMS
IF NUMERIC	IF VERIFY(FLD1,'0123456789')=0;	If a byte in the char fld FLD1 is not in '01…9' that byte's position number is returned
IF ALPHABETIC	IF VERIFY (UPCASE(FLD1),'ABCDEFGHIJKLMNOPQRSTUVWXYZ') =0;	Similar to above… also, the UPCASE function makes any lower-case chars into upper-case

COBOL PICs and SAS Informats (input formats)

COBOL PIC CLAUSE	SAS INFORMAT EXAMPLE	NOTES
		NUMERIC INFORMATS
PIC 9(5)	5.	The '.' Indicates to SAS that this number is an informat
PIC 999V99	5.2	Total length is 5 bytes, implicit or explicit decimal point
PIC S9(5)	ZD5.	ZD for zoned-decimal
PIC S9(5)V99	ZD7.2	Field is 7 bytes long,
PIC S9(5)V999	ZD8.3	Field is now 8 bytes, and last 3 digits are decimals
PIC S9(5)V99 COMP-3	PD4.2	4 bytes long, of which last 2 digits are decimals
PIC 9(5)V99 COMP-3	PK4.2	Same as above but unsigned (this is rare)
PIC 9(8) COMP	IB4.	Integer binary (signed)
	COMMA14.2	Reads a number 14 bytes long w/ commas, e.g. 204,399,871.45
	YYMMDD8.	Reads a YY/MM/DD input, converts it to a SAS (serial #) date
	YYMMDD10.	Reads a YYYY/MM/DD input, converts it to a SAS (serial #) date
	MMDDYY6.	Reads a MMDDYY input, converts it to a SAS (serial #) date
	TIME5.	Reads a HH:MM time, converts it to a SAS (serial #) time
	TIME8.	Reads a HH:MM:SS time, converts it to a SAS (serial #) time
	JULIAN5.	Reads a YYDDD date, converts it to a SAS (serial #) date
	BEST10.	SAS chooses the best notation for a 10-byte field
	BITS8.	Extracts bit values of a 1 byte field
	MRB4.	Microsoft real binary (floating point) number 4 bytes long
		CHARACTER INFORMATS
PIC X(5)	$5.	Reads 5 bytes of char data and trims off leading blanks
PIC X(5)	$CHAR5.	Reads 5 bytes of char data and does not trim off leading blanks
PIC X(5)	$CHARZB5.	Reads 5 bytes of char data and does not trim off leading blanks
	$VARYING110.	Reads a variable-length 110 byte field INPUT VARLEN DD1 XXX;
	$UPCASE200.	Converts lowercase char values to upper-case for a 200-byte fld
	$EBCDIC32.	Convert EBCDIC to ASCII for 32 bytes

Some COBOL PICs and SAS (Output) Formats

COBOL PIC CLAUSE	SAS FORMAT EXAMPLE	NOTES
PIC ZZZZ9	5.	The '.' Indicates to SAS that this number is a format
PIC Z9.99	5.2	Total length is 5 bytes w/ explicit decimal point
PIC 9(5)	Z5.	Print leading zeros
PIC ZZZ,ZZ9.99	COMMA10.2	Field is 10 bytes long, including punctuation
PIC $$$,$$9.99	DOLLAR10.2	Field is 10 bytes long, including punctuation
	BEST10.	SAS determines 'best' format for the data (in this case, for 10 bytes)
	WORD50.	Writes out numbers as English words (50 bytes)
	ROMAN20.	Writes out numbers as Roman numerals (20 bytes)
	SSN.	Writes out 9 digit number with dashes (11 bytes)
		DATE FORMATS
	YYMMDD10.	Writes a SAS date value as YYYY/MM/DD
	MMDDYY6.	Writes a SAS date value as MMDDYY
	TIME5.	Writes a SAS time value as HH:MM
	TIME8.	Writes a SAS time value as HH:MM:SS
	JULIAN5.	Writes a SAS date value as YYDDD
		CHARACTER FORMATS
PIC X(5)	$5.	Writes 5 bytes of character data, trims leading blanks
PIC X(5)	$CHAR5.	Writes 5 bytes of character data, including leading blanks
	$VARYING120.	Writes up to 120 bytes of depending on LEN option

… a simple SAS program compared to a COBOL program:

```
DATA _NULL_;                                        *** _null_ means don't produce a SAS output file;
    LENGTH FLD_A $3;                                *** like 01 FLD-A PIC X(3)
    INFILE IN END=EOF1;                             *** same as OPEN INPUT file;
                                                    *** AT END SET EOF TO TRUE;
    FILE OUT;                                       ***same as one-time OPEN OUTPUT file;
    DO UNTIL (EOF1);                                *** same as PERFORM UNTIL EOF WITH TEST AFTER;
        INPUT @1 ONE $CHAR5. @10 TWO $CHAR5.;       *** same as READ INTO;
         REC_CNT + 1;                               ***same as INITIALIZE TO 0 and ADD 1 TO REC-CNT;
        TEMP=ONE; ONE=TWO; TWO=TEMP;                ***same as three MOVEs;
        FLD_A=SUBSTR(ONE,2,3);                      ***similar to UNSTRING
        PUT _INFILE_ @1 ONE @10 TWO;                ***same as three MOVEs and a WRITE;
    END;                                            ***same as END-PERFORM;

   FILE LOG;                                        ***same as one-time OPEN OUTPUT LOG FILE;
   PUT REC_CNT=COMMA20.;                            ***same as DISPLAY (TO LOG (sysout));
STOP;                                               ***same as GOBACK or STOP RUN after 1st loop
RUN;                                                *** same as GOBACK or STOP RUN at end of pgm
```