ABSTRACT
SAS, as a programming language, has been in existence for over 25 years and its uses and applications are utilized in many different industries. One of these industries is the pharmaceutical industry. The pharmaceutical industry (including but not limited to pharmaceutical and bio-tech companies, contract research organizations (CRO’s) and a multitude of small, special interest companies) is a major and rapidly increasing user of SAS. The discipline of Clinical Trials within the industry is one of the largest users of SAS, by virtue of its’ need for intricate statistical analyses, reporting and summarization requirements, and the need for high quality output. So what makes programming in the pharmaceutical industry different from other industries? The FDA regulates the pharmaceutical industry.

INTRODUCTION
Since the pharmaceutical industry is a governmentally regulated industry, SAS programmers must adopt many unique philosophies and techniques in order to meet the requirements for high quality output. This presentation, for an audience of employees new to the industry or those just interested in the industry, will offer overviews of the following areas: industry standards and expectations, documentation, validation and good programming practices.

INDUSTRY STANDARDS AND EXPECTATIONS

STANDARD OPERATING PROCEDURES
SAS programming in the pharmaceutical industry is governed by requirements, guidances and expectations. As such, one key component that outlines a company’s standards and methods of ensuring that these standards are met is through the use of Standard Operating Procedures (SOPs). SOPs are an FDA requirement and the ICH (International Conference on Harmonisation) defines standard operating procedures as “Detailed, written instructions to achieve uniformity of the performance of a specific function.” How SOPs are organized is specific to each company, however they all cover similar issues – generally any process of specific interest to the FDA and/or mentioned in the Code of Federal Regulations (CFR). SOPs tend to be broad but it is key for every programmer to know what SOPs apply to their job function and to make sure those SOPs are followed.

SAS SOFTWARE
FDA requires SAS Version 5 compatible files (transport files created with PROC COPY, not PROC CPORT). What this means for programmers is that “old” data set, variable name, and variable attribute conventions must be followed. With the advent of SAS version 7 and beyond, long variable names, long labels, and text variables over 200 characters became available to programmers. Unfortunately, when data sets get converted to SAS Transport files, variable names get truncated to 8 characters, text variables get truncated to 200 characters, and labels get truncated to 40 characters. As a result, although more is possible, programmers should stick to SAS version 6 conventions when creating data sets.

CDISC
Although not a current requirement, FDA has officially adopted the Clinical Data Interchange Standards Consortium (CDISC) Study Data Tabulation Model (SDTM) for submitting clinical study data. CDISC standards are a set of documents that outline and define the structure of submission database data sets and the accompanying documentation. These standards are not an absolute, they are simply guidance. FDA does not require submissions in CDISC, but if used, submissions are expedited and easier to perform.

DOCUMENTATION
Documentation is one of the most important aspects of SAS programming in the pharmaceutical industry. Proof of the accuracy and precision of output created by programs must be maintained in the event of an audit. Because programs are often recycled across similar projects, documentation also serves to answer questions about what was done, how and why. This is key in an industry where the development of a product can take place over the course of many years. Documentation can be broken down into two categories: External documentation and internal documentation.

EXTERNAL DOCUMENTATION
There are several different types of external documentation that are required as part of the study files as a whole. Before programming can begin, the following documents should be collected as a resource for the programmer: the study protocol and any amendments, an annotated Case Report Form (CRF), Statistical Analysis Plan, meeting minutes and notes, company programming standards and any relevant SOP’s.
Once programming has begun, there are several types of documentation that programmers are responsible for producing – both as proof of validation and as valuable information. For validation, documents include: SAS programs, logs and list files; any relevant documentation (email information, analysis rules, and descriptions of “odd data”); validation checklists; “change control” documentation. Documents that are both required by the FDA and provide a good information resource include Data Definition Tables (DDTs) and program directories. DDTs are essentially PROC CONTENTS taken to one more level. This document includes detailed descriptions of each data set, including the variable name, a brief description of the content, the type, the length, any possible values, and a description of how the variable was created (for example, the formula used to calculate AGE). A program directory includes a list of all programs and the output they produce (both the file name and a brief description of what is in that file).

**INTERNAL DOCUMENTATION**
In addition to external documentation, programmers must also create documentation within the program itself. This is perhaps the most important documentation to create. This includes, but is not limited to, the following:

1) Program Header: The header of a program is a very detailed section of the program that describes creation, modification and change history. At the very least, the program header should include the program name, author, purpose of the program, version of SAS used, and any modifications made to the program after it was validated. For macro programming, the header should also include input and output variables and their uses. An example of a standard program header would look like this:

```sas
/* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * *
* Program:     QC.SAS
* Author:      Brian Shilling
* Created:     13APR2004
* Purpose:     SAS program to perform the QC of the SAS logs from study
*              programs.
*              
* Macros:      QC - Main macro to perform QC
*              QC1 - Run the actual QC process
*              
* Modifications:
*              B. Shilling - 20JUL2004
*              Added code to check the platform on which SAS is running
* * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * */
```

2) Program Body: Throughout the program code, the programmer should make sure to add comments and plenty of white space. Programmers should be sure to use comments to explain unusual or complex SAS code, add comments to visually separate main sections of program logic, and include any reference to external documentation that has a direct impact on the program. Generous comments in the code make it easier to know what needs modification if analysis rules change or the program gets used for another study.

**VALIDATION**
Validation is the act, or process, of proving the accuracy and integrity of the output of the programming being performed. All of the work a programmer does will be scrutinized by several entities: First and foremost, FDA; Clients, both internal and external; and programming peers. Too many mistakes found in programming output will undermine the client’s faith in it and lead to longer review time and/or tense relationships between parties. Validation in the pharmaceutical industry is a very important, required step.

**VALIDATION TOOLS**
Given that validation is required, what tools are available to programmers to be sure that validation is useful and accurate? The most important tool is the SAS Log. The SAS log will provide numerous and varied answers to validation questions. Make a list of all of the keywords that you will need to look for and be sure to scan the log for all of them. It is even possible to write a SAS program to read all of the .LOG files and scan for as many keywords as you would like. Be sure to review beyond the errors and warnings. Follow observations across data steps and look for unacceptable notes (automatic to numeric conversions, division by 0, etc.).
Other tools that are available to the programmer are SAS options. System options such as MPRINT, MLOGIC and SYMBOLGEN (SGEN) are useful tools that will list macro variable resolution and structure. MSGLEVEL is also a very useful tool. If MSGLEVEL is set to the letter i, the log will contain information for any merge statements that have the same variables in each data set being merged (other than the variables in the BY statement). Any variables that have their values overwritten during a merge will be listed in the log. In addition to SAS system options, using the IN= option on data sets when merging is another useful tool. This option allows programmers to keep track of where data is coming from and allows for control of the data based on where it is coming from.

TECHNIQUES TO FACILITATE VALIDATION

It is very important when programming to make sure that the data you are working on meets certain basic assumptions. Programmers need to be familiar both with what the data is supposed to be and what it actually is. Discrepancies between assumptions about the data and reality can lead to either revision of the data or revision of the analysis plan. To this end, it is important to validate any data assumptions and there are various ways to do this.

1) Perform frequencies on categorical variables to make sure there aren’t any unexpected categories (for example, GENDER should be a 1 or 2 but the data contains missing values).
2) Use formats to get frequencies of continuous variables wherever appropriate. For example, if AGE should be between 18 and 65, create a format where VALUE 18 – 65 = ‘Good age’ other = ‘Bad age’ and apply that format in your PROC FREQ statement.
3) Print select cases before manipulating the data so that a starting point is solidified and a baseline for later checking is established. Print the same cases at the end of the program and compare the results.
4) When possible, use DATA steps rather than PROC SQL so that validation checks can be programmed in as inherent components of the program.
5) Use multiple data sets instead of the DELETE statement to see what records are being discarded. Put the records you want to keep into the target data set and put the discarded records into a “dropped” data set and print the dropped records to make sure the correct records are discarded.
6) Use FIRST(dot) and LAST(dot) processing rather than NODUPKEY and put the dropped/unprocessed records into a checking data set that can be printed for review.
7) Use tools like DATA _NULL_ or macro variables to print conditionally.
8) Use PROC FREQ with the LIST and MISSING options on the TABLE statement to check data type conversions and decode variables. For example, if GENDERCD (values of 1 and 2) is decoded into the variable GENDER (values Male and Female), use PROC FREQ with TABLES GENDERCD*SEX / LIST MISSING to ensure that all values were decoded correctly.

VALIDATION TIPS AND TIMING

When planning a validation process, consider the following tips:

1) Plan ahead – how much programming is required and how much time is available?
2) Start programming data sets before the database is locked – data issues that were missed by data management missed or didn’t account for.
3) Program the corresponding data listing(s) as soon as possible.
4) Create the data definition tables prior to validation
5) Scrutinize everything and ASK QUESTIONS
6) Follow the validation checklists to the letter for all output.
7) Every data set needs to be completely validated – NO SHORT CUTS!
8) Every piece of table/listing/graph output needs to be validated, but it is possible to limit the checking to the following:
   a) “Cosmetics” – spelling in titles and footnotes, general presentation, pagination, etc.
   b) All “key” numbers such as population counts and p-values
   c) Check a “reasonable” percentage of everything else.
   d) If one program generates several tables, check the first thoroughly and then only a portion of the remainder.
   e) For changes made after validation, check the items that changed but spot check other items to make sure you didn’t change something unintentionally.

GOOD PROGRAMMING PRACTICES

MAINTAINABLE CODE

All programming code must be maintainable and be simple and programmer friendly. All programmers should follow meaningful program, data set and variable naming conventions. Each program should be easy to read and do only one thing (create 1 data set, 1 unique table or 1 unique table with similar subsets.). Programmers should keep in mind that drug programs take years to complete. Questions regarding data and code come up long after a study is
completed. Programs must be easily maintainable and understandable so that future programmers and easily visit the programs.

MAINTAINABLE DATA
In addition to the study programs, all study data must be maintainable. Data sets should be stored with variables in a logical viewing order and sorted by variables that create unique record identifiers. All variables and data sets should be labeled and only keep the variables needed or required. Because data are often times submitted to outside entities, programs should not contain any non-SAS formats. Do not hard code any data changes in the programming, all changes should be made through the proper channels.

CONCLUSION
Due to government regulations, SAS programmers in the pharmaceutical industry may have to relearn their programming skills or learn new ones. This presentation touched on several key programming areas: industry standards and expectations, standard operating procedures (SOP’s), documentation, validation and good programming practices. These ideas and practices will make pharmaceutical programming much easier and complete when submitting a clinical study.

ACKNOWLEDGMENTS
Many thanks are extended to Carol Matthews for her work and assistance with this presentation.

CONTACT INFORMATION
Your comments and questions are valued and encouraged. Contact the author at:
Brian C. Shilling
Director of Quality Assurance
Applied Clinical Intelligence
212C Race Street
Philadelphia, PA 19106
Phone: 215-627-2100 X19
FAX: 215-627-1110
Email: bshilling@a-ci.com
Web: www.a-ci.com

SAS and all other SAS Institute Inc. product or service names are registered trademarks or trademarks of SAS Institute Inc. in the USA and other countries. ® indicates USA registration.

Other brand and product names are trademarks of their respective companies.