THE IMPORTANCE OF DOCUMENTATION
Metropolitan Edison Company

ABSTRACT
The theme of this paper, as you can tell by the title, is the importance of documenting your SAS programs and systems. In this paper, I will explain why I believe documentation is necessary and in some cases mandatory.

INTRODUCTION
Before I get too far along, I want to point out how we have been using the SAS system. The SAS system is a huge system and serves a number of different functions. At the Metropolitan Edison Company Information Center, we use SAS as a programming language to produce ad hoc reporting from mainframe systems and as a programming language to develop departmental decision support systems. This paper stresses the importance of documentation in reference to those two areas. However, the documenting facilities of SAS are not limited to those areas. The concepts presented in the Documenting Your Program section work with any PROC or DATA step coded.

The following sections will tell you who should document, why you should document, and how you could document your programs and systems.

WHY YOU SHOULD DOCUMENT
Documentation is not one of the fun programming tasks. Many programmers hate to do it. Most programmers would rather get to the next project. Managers think it's wonderful. In fact, the managers know it must be good because programmers hate it so much.

Documentation helps to preserve a sense of continuity in the midst of change. Applications are getting more complex as the proficiency of the end user grows. As the programmer becomes more proficient, systems are developed with subtleties that no one, not even the programmer, remembers if not well documented.

Increasingly, department applications are supporting important business functions and are therefore strategic to the operation of the business. It makes sound business sense to have these systems well documented. Documentation should be considered mandatory if the following conditions prevail:

BUSINESS IMPACT. Applications should be documented if they assist in business decisions or directly influence the output of a department or organization.

INTERACTION WITH OTHER SYSTEMS. Applications that share data or feed data to other systems should be documented.

DURATION OF USE. Watch out for the one-time-only applications. These applications have a way of becoming full-time permanent systems. If an application is used several times it's almost always worth running more than once.

VOLATILE APPLICATIONS. How many applications are being developed in your organization that make no sense to anyone but the person doing the programming? How often do you have people running programs without understanding what the program is doing because of inadequate documentation? Applications that must be revised frequently to meet changing business conditions should be documented. Each change should be incorporated into the documentation so that the end user can understand the current version of the system.

WHO SHOULD DOCUMENT
The SAS SYSTEM is a powerful system for information management. It provides an excellent platform for the easy development of decision support systems. Using the database management facilities and Full Screen Product, the SAS SYSTEM makes building systems so easy that it puts system development into the hands of the end user. The end user is in the position of controlling his/her own information resources.

The SAS System enables the end user to build complex applications with a wide spectrum of users. At the Metropolitan Edison Company information center, we use SAS as a programming language to produce ad hoc reporting from mainframe systems and as a programming language to develop departmental decision support systems. This paper stresses the importance of documentation in reference to those two areas. However, the documenting facilities of SAS are not limited to those areas. The concepts presented in the Documenting Your Program section work with any PROC or DATA step coded.

The following sections will tell you who should document, why you should document, and how you could document your programs and systems.

WHEN SHOULD YOU DOCUMENT
Just when the IS/MIS departments had their standards written for program development, system development, etc., the SAS System and other 4th generation languages opened system development to a whole new class of programmer. All the hard lessons learned by IS/MIS departments, now, need to be learned by a whole new group of computer users.

I don't want anyone to think that I'm against the end-user programmer or that I'm for the old line IS/MIS professional programmer. I don't believe computer programming is for the Information Services professional only. The SAS System blurs the distinction between the professional and the end-user programmer - we both use the same SAS System. I've seen excellent programming work done by end users that could put a lot of professional programmers to shame. I've also seen a lot of end user written code wasted that is what I want to see avoided.

The orphaned application is a good example of a situation to avoid. An accountant takes the SAS for Non-Programmers course and later the Advanced SAS course. He reads the SAS user's guide to cover after cover. Now having demonstrated a talent for programming, he becomes the department data processing expert. Before you can say "Free Tabulator", he's cooked up a few hundred SAS programs that everyone and everything in the department now depends on.
His programs have streamlined the month-end closing process so that what had taken days now only takes a couple of hours. The whole manual process has been automated. The department manager is thrilled.

At first glance, the manager thinks the department is running well. What he doesn't realize is that the department's history, in an effort to reach a quick solution to get something up and running fast, took shortcuts in both development standards and documentation. He has developed programs and systems that are totally undocumented and in so doing he has compromised the integrity and accuracy of the systems. You can all see the potential for disaster. Organizations are always in a state of flux. People get promoted, transferred, or resign and there goes the developer of your system. The only one in the department that understood the system. The only person who knew all the pieces were held. This scenario can have several different endings.

The person who takes over for the departed developer can't make any sense of the system and tries to rewrite the whole thing from scratch or the department manager calls the MIS department or Information Center to streamline their collection of programs or add a front-end to their system. I'm sure this sounds familiar to anyone who works in an information center.

A system was never developed, a bunch of programs just sort of evolved into a system, and for a time it worked. The manager knew there had been some problems, but the "hero" (the now departed developer) always got the system to work. It's only now at crunch time that the manager is interested in doing it right.

Another situation to avoid is the "never ending" project. I worked at one company where there was a constant stream of contracted employees. Every time a new contractor started work in a particular department, they were assigned the task of finishing work on a + customer information mailing labels system.

The new guy can't or doesn't want to figure out what his predecessor was doing, so he throws out the old code. Besides, he has a better way to do it. The old programs get dumped and we expanded more resources to get us back to where we've already been.

These problems can be avoided, if a few guidelines are followed:

- Identify who is developing software, and what they are developing. Supervisors of end user programmers should be actively involved in the software development process.
- Determine the scope of the application. Review the problem statement with your Information Center or MIS department. A program or system may already exist to fill your needs.
- Educate the people in your department of the dangers in an Orphaned Application.
- Adopt a common company standard for software development.

* Use the IC/MIS staff as a consultant to oversee the design process and see that development is in compliance with standards. Supervisors of end users who program should be aware of the risks and be committed to helping control them.

HOW TO DOCUMENT

Forcing developers to produce documentation can result in documentation that is done in the most minimal way. The value of documentation is only realized if the documentation is well done, if it is done poorly it is worse than no documentation at all. The only hope for producing good documentation is to convince the developer that it will benefit him to do so. If the act of producing documentation has no value to the developer, the job will always be done in a minimal manner, but if producing good reasonable documentation has demonstrable good effects, there will be no way of preventing good documentation.

Ultimately, the only way to guarantee documentation of these departmental decision support systems is to have department management take responsibility for the computing integrity within their department.

DOCUMENTING YOUR PROGRAMS

Program documentation starts with the program header and includes command lines and line-by-line comments adjacent to SAS program statements. The program header documentation should always provide the name of the program, the author, the date the program was written, and a description of what the program will do. Additional header information may include the names of data drivers, information about input and output files used by the program, project control information, and anything else that might be helpful to the next programmer that has to maintain this program. A sample of header documentation is shown in Figure 1.0.

Figure 1.0.

---

**SO005831**

**AUTHOR: S.VOYNAR**

**DATE: 10/12/88**

**PROJECT NUMBER (WEIS): 5033**

**PROJECT DESCRIPTION: VIGIL RETIREMENT AVSD AUTOMATION**

**BATCH JOB: R498000000**

**SAS MEMBER DESCRIPTION:**

```
* SAS PROGRAM CREATE THE TRANSACTION CARDS FOR INPUT INTO THE VIGIL CAPITAL RECOVERY SYSTEM.
```

**DATA FILES:**

* INPUT:
  - DNAME-BALANCE FILENAME SOURCE:JOHN1
  - DNAME=PROPERTY FILENAME SOURCE:JOHN1
  - DNAME=CONV FILENAME SOURCE:JOHN1

* OUTPUT:
  - DNAME=TRANSFILE FILENAME

**NOTES, CHANGES, ETC.**

---

843
Another important piece of program documentation is the comment lines in the program. I try to use comment lines to clearly indicate important sections of the program. In a program that reads a file once but creates many reports, I would use comment lines to show each report. That way, it's easy to find the source code you want. Figure 1.1 shows a program fragment with comment lines.

```plaintext
/* THIS SECTION WILL SELECT ONLY DRAWINGS */
/* AND KEEP THE REVISION */
IF INDEXING,*
   1) = 0 THEN OUTPUT; ELSE DELETE
KEEP INDEX DRAFTS NAME;
/* SORT THE AARAB DRAWING DATA BY THE DRAWING FIELD SO THE */
/* FILE CAN BE MERGED WITH THE DATA FROM THE ECN DATABASE */
PROC SORT DATA=AARAB BY INDEX DRAFTS NAME DESCENDING NAME;
```

The final piece of program documentation within the program is line-by-line comments using /*
Line-by-line comments are useful for documenting exactly what each line of code is doing. This is especially useful around "IF" statements. If you are including records only where the account code is equal to 32000, use a line-by-line comment and tell us that account code 32000 is an Operating & Maintenance account. This type of information is useful in and out of the program. Line-by-line comments are best done when you are coding the program. I know that if I don't do it at that time, I'll seldom go back and do it. Figure 1.2 shows the use of a line-by-line comment.

```plaintext
/* THIS ROUTINE OF THE DASD OCCUPANCY BILLING SYSTEM PROCESSES */
/* THE CURRENT MASTER FILE, REMOVING INACTIVE FILES CURRENT */
/* SPACE ALLOCATED = 0) AND APPENDING THEM TO THE CURRENT */
/* MONTH-TO-DATE INACTIVE ARCHIVE FILE */
/* */
/* INPUT: CURRENT DASD OCCUPANCY MASTER */
/* AND INACTIVE ARCHIVE FILES */
/* OUTPUT: UPDATED DASD OCCUPANCY MASTER */
/* AND INACTIVE ARCHIVE FILES */
/* */
/* WRITTEN BY: K. C. JONES DATE: 15 SEPTEMBER 1986 */
```

DOCUMENTING YOUR SYSTEMS

The Metropolitan Edison Company Information Center has been quite busy building departmental decision support systems. Because these systems are spread through out the company and are busy performing support services daily, special care has been taken to ensure their reliability. Key points of our system development documentation are:

1. System Overview
   - Narrative of system processing
   - Perceived benefit of system
   - Application Coordinator/key user information

2. User Documentation
   - Flow diagram of system processing
   - Logon through Logoff procedures
   - Screen layouts and options
   - SFF Panels
   - Input formats and processing

3. Technical Documentation
   - Security requirements
   - Fully qualified data set names for programs, clients, SPF panels, databases, and SAS/AF screens
   - Storage layout description for all databases
   - Special processing considerations
   - Error processing considerations

It is the responsibility of the assigned project analyst to complete the system documentation. This documentation must be in the standardized format outlined below.

Contents

1. System Overview
   - Narrative of system processing
   - Perceived benefit of system
   - Application Coordinator/key user information

2. User Documentation
   - Flow diagram of system processing
   - Logon through Logoff procedures
   - Screen layouts and options
   - SFF Panels
   - Input formats and processing

3. Technical Documentation
   - Security requirements
   - Fully qualified data set names for programs, clients, SPF panels, databases, and SAS/AF screens
   - Storage layout description for all databases
   - Special processing considerations
   - Error processing considerations

All decision support systems will be developed in SAS

All SAS programs will be moved to the company SAS code library. Read-only access to the data set is given to all Metropolitan Edison company users. Update access is granted to information center personnel only. Movement of programs is quick and easy, yet the programs are afforded protection from the general employee population.

An entity control system is maintained providing information about each SAS program in the company SAS library. This system also provides a unique id number for each program. This identification number is included in the banner documentation of the source code.

* The program identification number is included as part of the report heading. This reduces the time spent tracking down a program for a maintenance enhancement. See Figure 2.1. This example also includes the client drivers in the report heading.

844
All clist program drivers are to be loaded into the company department library for clists. Clist information is also stored in the entity control system. AD SAS formats used within Information Center developed systems are to be loaded into the company format library. All SAS/JAF menus, programs, and help facilities are to be loaded and executed from the company AF library.

THE EXCEPTION

For small systems, anything that takes less than 8 hours to build, we use an option of our entity control system, relevant to a small system, where names, variable names, screen catalogs, etc. are inputted to a screen as shown in Figure 5. A program produces the condensed documentation. The program source code is listed in Figure 3. The purpose here is to produce documentation that has value and a positive cost-to-benefits ratio.
As you can see, with documentation you can go to different depths. You can go from minimal effort to something which could rival War & Peace. If your situation is such that you can’t produce highly detailed documentation, a minimal effort is better than nothing. The quality of your coding, no matter what level you program, is improved with some documentation. Your goal is to produce documentation that has value not necessarily volume.