Interactive Applications (CANDA) Development with SAS for Windows
Frank S. Palmieri, Merck Research Laboratories
Rob Rosen, Merck Research Laboratories

Introduction
SAS for Windows (6.08 and 6.10) can be used to develop sophisticated interactive applications on a PC. However, there are many details that must be addressed when constructing a successful application in this environment. These issues span hardware, operating systems, and software.

In developing a Computer Assisted New Drug Application (CANDA) for the pharmaceutical industry, our team has made extensive use of SAS for Windows. Many of its features are new to the SAS System and are specific to the Microsoft (MS) Windows environment. The focus of this paper is on the advantages and drawbacks of development with SAS for Windows.

Hardware topics include selection of processor, hard drive, memory, and BUS as related to application response time. Operating System (OS) issues include MS-DOS and MS Windows with regard to application performance and reliability. Software elements cover Screen Control Language (SCL), the FRAME entry, and object oriented programming (OOP). Other aspects include Microsoft "look and feel", dynamic data exchange (DDE), and SAS for Windows limitations as a client/server application.

Background
A CANDA is an interactive document and/or data review tool which can include robust functions for accessing and analyzing drug related information. Pharmaceutical companies provide these applications to regulatory agencies (i.e., the Food and Drug Administration) to expedite approval of their newest drugs.

SAS for Windows provides the ability to create an interactive Graphic User Interface (GUI) for an application. It also supplies powerful features for accessing and analyzing data. This makes SAS for Windows a well-suited tool for developing interactive applications like a CANDA.

Hardware Issues
Microprocessor selection is an important choice when developing resource intensive applications. When considering 486 Central Processing Units (CPU), a faster clock speed will increase response time. However, a 66 MHz 486 DX2 chip will not necessarily double the program execution time of a 33 MHz 486 DX chip.

A Pentium (586) processor may or may not be faster than a 486. Pentiums are 64-bit based and can process more information with each CPU cycle than a 32-bit 486. However, software must be specifically written to utilize 64-bit technology. At present there are no 64-bit PC operating systems. Therefore, a PC cannot exploit the full potential of a Pentium processor.

A platform specific issue that can substantially increase system performance is the selection of internal BUS architecture. The four choices are ISA, EISA, VL BUS, and PCI and are 8, 16, 32, and 64-bit architectures, respectively. PCI is presently the fastest of these standards. Although applications and operating systems may not be written to use a 64-bit environment, the internal architecture will effect the overall response time.

Memory is an extremely important component in the design of a PC platform. When an application runs out of memory, it will use the hard drive as "virtual memory" and begin paging/swapping. Information exchange is much slower between CPU and hard drive than between CPU and memory. The amount of memory to use is application specific. (Note: SAS for Windows functions best with a minimum of 16 MB.) For example, a dataset that is 15 MB can completely sort in memory if you have more than 15 MB of memory in your machine. (Note: The SORTSIZE option in the CONFIG.SAS file must be set to 15MB or larger.)

Selection of a hard drive can increase system response time. The important performance characteristics of hard drives are seek and scan rates, throughput, and burst rate. Lower seek and scan rates and higher throughput and burst rates are better. The two most common types of hard drives are IDE and SCSI. Seek and scan rates on these drives are similar, however, SCSI drives have higher throughput and burst rates.

SCSI drives are typically easier to setup and have larger capacities than IDE (up to 4 GB or more), but are more expensive and require a SCSI controller. Some SCSI controllers allow for hardware cache which can greatly increase performance in an I/O intensive application.

Another feature of hard drives is 32-bit disk access (as supported by MS Windows for Workgroups). This will improve I/O between disk and CPU. Drives must be Western Digital Standard compliant to use the 32-bit disk access driver (WDCTRL). There are currently no SCSI hard drives that allow 32-bit disk access.

OS Issues
Choosing an operating system/GUI interface is an important decision as well. MS-DOS with Windows is currently the most predominant PC configuration in the user community. This does not necessarily mean it is the best configuration.

MS-DOS with Windows is known for yielding General Protection Faults (GPFs). Applications running under MS-DOS use an unprotected memory area. If a program was written improperly (i.e., contains a hardcoded memory address), it may occasionally access the same memory area as another program. This will result in the application freezing up, a GPF, program termination, or any combination of the above. GPFs are intercepted and handled better, but will still occur in the newer versions of MS-DOS and Windows. GPFs are not nearly as common when using an operating system like OS/2 or Windows NT. Upgrading to Windows 3.11 or Windows 3.11 for Workgroups (WFW) results in improved memory management and provides 32-bit disk and file access (which improves I/O between disk cache and CPU).
Both Windows 4.0 ("Windows 95" -- not in production at the time of this publication) and Windows NT replace Windows and MS-DOS. They are self-contained operating systems with GUIs, offer a protected memory area similar to OS/2, have easier memory management, and no RAM limit (such as the 64 MB memory limit in MS-DOS). Only software specifically written for Windows 95 and Windows NT will use protected memory A, and will be prone to GPFs. The Windows 95 GUI will bear little resemblance to the current Windows look and feel. This version of Windows will be similar to the Apple MacIntosh front end with folders instead of groups and icons.

Applications written for a 32-bit environment can run on Windows or WFW (both 16-bit), but must have a 32-bit emulator (WIN32S) installed. Applications will run in a 16-bit mode. Windows NT, a 32-bit environment, will run these applications in a true 32-bit mode.

SAS for Windows

In the MS Windows environment, careful consideration must be given to the "look and feel" of your application. "Look and feel" is the way objects (i.e., pushbuttons, listboxes, etc.) appear and function on the screen. Most Windows products follow MS standards for "look and feel". Packages that do not adhere to these standards can be clumsy for an experienced Windows user.

SAS for Windows 6.08 is not totally compliant to MS standards. For example: not all FRAME objects are three dimensional, mouse/keystroke combinations do not function properly within FRAME objects, screen colors cannot be changed using "Control Panel", printing bypasses "Print Manager", and help does not use the Help Engine. Most of these items have been addressed in SAS for Windows 6.10. However, applications developed using FRAME still use non-standard objects (i.e., control arrows and icons) and mouse/keystroke combinations do not function to MS standards.

When using SAS for Windows, interaction with other software (i.e., MS Excel) can be important. Dynamic Data Exchange (DDE) is one form of communication between packages. It is a method of sending and receiving data/commands. "Dynamic" implies an "active" DDE link is set up between two applications. When information is changed in one application window, updates are automatically reflected in the other application window.

There are two types of DDE applications. A DDE "client" can read and send data to a DDE "server". However, it cannot receive commands. In other words, it can use another application, but the other application cannot use it. A DDE "server" can read and send data as well as receive commands. It can use another application, and the other application can use it.

SAS for Windows 6.08 is strictly a DDE client application. Other Windows applications can be invoked from the SAS System, the contents of SAS datasets can be passed, and commands can be sent to perform operations on this data in the other product. However, there is no way to notify the SAS System to read the data back, or to perform any "dynamic" changes to this SAS dataset based on what has occurred in the other application.

With SAS for Windows 6.10, there is limited DDE server capabilities. This is implemented through the Microsoft Windows Open Database Connectivity (ODBC) standard. The SAS System supplies a "database" server engine which allows other Windows products to access and operate upon SAS datasets. However, specific SAS commands still cannot be invoked from other Windows applications. Therefore, you cannot easily take advantage of SAS System features from other Windows packages.

Design

SAS for Windows provides the ability to create a GUI for your customized applications through an extension to SAS/AF called FRAME technology. FRAME allows for the creation of screens with Windows objects (i.e., pushbuttons, listboxes, checkboxes, radiobuttons, etc.). Through popup menus and "drag and drop" mouse movements, one can create a screen (frame) by choosing and positioning objects. Once an object has been placed, it may easily be relocated by clicking, dragging, and dropping with the mouse.

Also, with the advent of FRAME technology, Object Oriented Design (OOD) is now possible with SAS for Windows. OOD is a methodology for building applications where code revolves around "objects". Objects can be screen items like pushbuttons, listboxes, or entire screens themselves. Objects can also be data structures like an array, a linked list, a database table, or even an entire database.

The underlying principle of OOD is any code that manipulates an object is bundled with the object "class". An object class is a generalization for a type of object. For instance, a pushbutton is an object class. An actual object (object "instance") is an OK button or a Cancel button. Separate code modules for an object class are called "methods", and all methods reside together. Code that operates on an object cannot be written in external modules. Furthermore, anyone writing external modules cannot access the code behind the object classes' methods. They may only call these methods to manipulate an object. (Note: This is called encapsulation.)

FRAME technology provides an environment for developing and encapsulating methods with screen objects. However, it does not provide this facility for data structures. Therefore, FRAME is not a complete object oriented development tool.

Consequently, OOD with SAS for Windows is possible when designing the interface portion of an application. This may be achieved by first "mocking up" your application screens based on the object classes which are available with FRAME. Next, find common objects across screens. These will become application specific object classes. Also, examine functionality that may be required to manipulate common screen objects. These will become methods that can be tied to each object class.

FRAME comes with a screen object class resource list. This is the list of all object classes that are available when creating a screen. New object classes (subclasses) may be created from
these pre-existing classes. For instance, if your application uses an OK button on almost every screen, you may create an OK button object subclass from the pushbutton object class, and place it in the resource list. The OK button is then immediately available for use on every new screen.

Each object class in the resource list has a number of methods which manipulate it. Methods may be added to an object class by referencing new code in the object classes' method list. For example, if your application needs to save screen settings whenever you press the OK button, you may design and add a "save screen settings" method to the OK button class.

Prototyping
Another advantage of FRAME technology is the ability to perform Rapid Application Development (RAD). RAD is a methodology for quickly developing systems with dynamic user interaction. Effective implementation of RAD is dependent on quick changes to your interface and functionality.

FRAME's ability to create screen objects, relocate them, and change their attributes through drag and drop, point and click mouse manipulation makes RAD practical during screen development.

Screen navigation functions may be implemented with little if any code. However, quick changes to back-end processing are not as simple. Therefore, RAD may be applied more effectively during prototyping where the interface and system requirements are being defined. Full scale functionality may be built in later during development.

Development
SAS for Windows presents many options to be considered for application development. An initial decision must be made between using an OOP or procedural approach. Other issues include choice of screen objects, data structures, and coding languages. All of these decisions can affect the performance and the maintainability of your application.

An OOP approach to development requires more "up front" design time than the standard procedural method. An application must be viewed in its entirety to determine which objects, both screen and data, are being used throughout the system. Methods must then be designed to manipulate these objects. Once this has been completed, development can begin.

The advantage of the OOP approach is the development of neatly packaged, reusable object classes. The disadvantage is the investment of time before actual coding begins. In the procedural approach, many lines of code may be written in the time that it takes to plan your object classes and associated methods. This is clearly a decision which must be made based on the life expectancy and reusability of your application.

An SCL list resides in memory and may be created, populated, and deleted during execution. It dynamically utilizes as much memory as it needs at run time and relinquishes it when no longer necessary. (Note: This is an advantage over an array which allocates a predefined amount of memory.) An SCL list may be stored as SLIST catalog entries. Also, an SCL list can be used to populate a listbox through the object's attribute screen.

An SCL list that contains one column of data is easy to manipulate. However, SCL lists can be composed of sublists that contain multiple "columns" of information. Manipulating this type of structure is not as intuitive and requires careful design to avoid confusion. Managing multi-column data is more straightforward in a SAS dataset structure. The contents of an SCL list must be examined and manipulated through code whereas a SAS dataset may be browsed and edited through the FSVIEW procedure. Finally, searching through a large SCL list can take considerable time. This is because the nature of its structure an SCL list must be read sequentially. A large indexed dataset is faster to search through than an SCL list of equal size.

Although methods or programs associated with FRAMES are written in SCL, Base SAS language and procedures may be used through an SCL submit block. This feature is important in terms of performance as dataset processing is generally quicker when using Base SAS instead of SCL. An explanation for this is the extra time required to transfer data back and forth between the Data Set Data Vector (DDV) and the SCL Data Vector (SDV).

Dataset processing may be performed using either SAS data steps or the SQL procedure. In most cases the use of these "languages" is interchangeable. There are instances, however, where only one of these languages will provide the necessary
functionality. A SAS data step, for example, can be used to create multiple output datasets in one pass while SQL cannot. In turn, SQL can be used to perform a many-to-many merge, or join, between datasets to form a cartesian product while a SAS data step merge cannot.

Another difference between these languages is that SAS data steps are generally faster than SQL. One of the main reasons for this is that SQL attempts to sort input datasets before merging them, even if they are presorted. Specifying SORTEDBY variables for input datasets will remedy this problem. SQL code should then execute as fast as the equivalent SAS data step.

Summary

SAS for Windows provides the tools for creating sophisticated and robust interactive applications (like a CANDA) on the PC platform. However, the choice of hardware and operating systems is critical to the performance of any software that is built upon them. Also, to effectively employ SAS for Windows as a development tool, one must learn to utilize native Windows features and new programming techniques unique to this environment.

Selection of appropriate hardware components will improve application response time. When selecting a microprocessor, note that there is not always a linear relationship between clock speed and program execution time. Also, a Pentium processor may or may not be faster than a 486. Less paging/swapping will occur with more memory. The amount of memory to use is application specific. PCI is presently the fastest internal BUS architecture. With regard to hard drives, lower seek and scan rates and higher throughput and burst rates are better.

When considering Operating Systems, MS-DOS with Windows is currently the most predominant PC configuration in the user community. Applications running under MS-DOS use an unprotected memory area and can yield GPFs. Windows 95 and Windows NT, which replace Windows and MS-DOS, will use protected memory. Windows 3.11 and newer versions provide 32-bit disk and file access which improves performance. 32-bit applications will run in a 16-bit mode under Windows and WFW (by installing WIN32S) and will run in 32-bit mode under NT.

SAS for Windows can be utilized to create a graphical user-friendly front-end, to aid in quick prototyping, and to support OOD. Although version 6.10 generally follows Microsoft standards for "look and feel", FRAME AF applications developed with this product do not completely adhere to these standards. Also, version 6.10 is a DDE client application with limited DDE server capabilities. SAS for Windows provides the ability to create a GUI for FRAME applications. Furthermore, it facilitates object oriented design and development of the application interface. However, it does not promote this approach where design and development of background processing is concerned. SAS for Windows also facilitates RAD of an application's front end. However, it does not support the RAD methodology on the back end. Therefore, RAD with SAS for Windows should not be applied when creating a fully functional prototype.

FRAME technology coupled with SCL, Base SAS, and SQL presents many options during implementation. FRAME provides numerous predefined screen objects. However, when two types of objects perform the same function (i.e., listboxes and extended tables) it may be more advantageous to use one object versus the other. SCL supports linked list data structures which can be very useful in conjunction with FRAME technology. However, linked lists should not exclusively be used because they are easier to manipulate. Although methods or programs associated with FRAMES are written in SCL, Base SAS language and procedures should be utilized for dataset processing to enhance the performance of your application. Depending on the situation, datasets should be processed using a mixture of SAS data steps and SQL.

Acknowledgments

Special thanks to the following people for their input and support to this CANDA project:

Wanda Bidlack
Barry Cohen
Andrea Contino
Merck Research Laboratories
Planning Data Systems
Contino Associates, Inc.

Trademarks

Microsoft, Windows, Windows for Workgroups, Windows 95, Windows 4.0, Windows NT, WIN32S, MS-DOS, Word, Excel, are a registered trademark of Microsoft Corporation, Inc.

SAS, SAS for Windows, SAS/AF, FRAME Technology, SCL, Base SAS, FS VIEW are registered trademarks of SAS Institute, Inc.

Pentium is a registered trademark of Intel Corporation, Inc.