The format used for this morning's panel discussion is to discuss the progress we have made in developing the SAS System for Minicomputers over the last few months because we realize that most of you who are using the portable SAS System are using Release 4.03 or 4.04, and we have just recently shipped 4.06. There have been significant changes made between these two releases, and we feel that in order to ensure that everyone understands what we are talking about versus what you are probably familiar with, we will describe the difference, or at least some aspects of it. Then, we are going to talk a little about what we have planned to develop over the rest of this year; essentially in a very general format, but we will at least try to answer a lot of your questions before they come up. That is basically the presentation. We are going to have two of the developers talk about the progress, and each one will talk about the prospects in his particular area.

To start off, let me introduce each of the developers:

- Bruce Tindall is manager of the Portable Systems Compiler Project.
- Darrell Massengill is manager of the Host System Interface for the Data General and the Prime Systems. Darrell is also looking at other machines that we may want to use in the future.
- Steve Beatrous is manager of the Supervisor and Run Time Interpreter Group—the integral interface between the Compiler and the Host Level people.
- Mark Schaffer is a previous member of the VAX Host Development Team. Mark has recently moved to the Compiler Development Team.

We do not have the PROC interface present. I think that you are aware that much of this Conference is aimed at different applications of the procedures and different views of what the procedures do. Let me emphasize that the procedures we use are developed by the same people and are essentially the same procedures that are used on the IBM side, and that is the common tie right now between the two systems. So, when we talk about the progress and prospects that we make regarding the portable SAS System, we are talking about the DATA STEP portion of the system which is the compiler, the interpreter, the supervisor and the host system interface and not necessarily applicable to the procedure. In this regard, let me make a couple of comments.

- The Display Manager is fully implemented on the system and is available for your view in the demonstration area. We have and will continue to implement all of the SAS products on the portable system during the coming year, but our major emphasis this year and during the immediate short term is to improve the general performance of the system by making changes to the system itself and by putting more of the code into machine language. That is where our emphasis is, and we feel that is where you want it to be.

- On the subject of debugging, we know of about 80 bugs that are in the system right now. We think that is rather amazing, spectacular, and a salute to the effort that we have gone through to bring you a system that is solid. With each release, we hope to keep the number of bugs within the system under 100. If you want to make any comments on this, please let me know. Also, if you think that we should give something higher priority than performance and bug fixing in the system, please let me know, and we can talk about it at some other time and talk also about the philosophy of our development.

Bruce Tindall will start the discussion with the progress that we have made, then he will immediately be followed by Steve to finish detailing the progress of the compiler and the supervisor, but he will also include the other aspects.

Bruce Tindall:

I would like to talk about some of the progress we have made since Release 4.04 for VMS/SAS and Release 4.03 of the AOS/VS SAS beta test releases. Since then, we have implemented quite a few new features and have done quite a bit of work that is now available in the production release that has been shipped to VMS/SAS sites already. I am going to talk about a few things that have been added to the PROC step in general, some new features that you will see on your SAS log output, and a couple of new miscellaneous features.

You may have noticed that you could not specify negative numbers or missing values in certain PROCs before. Now you can. This is especially useful for things like specifying axes in PLOT and GLOT; you can specify negative numbers along the axes and use PROC EDITOR to put missing values and negative numbers into the SAS data sets that you are editing.

The BY statement in PROCs now accepts the NOT SORTED option. This can save you some time when you know that your input data set to a PROC is ordered but not sorted; you do not have to sort it now to use BY processing in the PROC because you can say:

```
BY variable name NOT SORTED;
```
When PROCs read unusual data sets like data sets with no observations or data sets with very large numbers that caused overflows, you often got an abend in the earlier versions of the SAS System. These are all taken care of now. You get a SAS error message and then processing continues. Since Release 4.04, the PROCs that have had the most bugs fixed in them have been EDITOR, BROWSE, SORT, and CALENDAR. So, with all aspects of the system, but especially with those PROCs, if you had any problems before, please try them again in the new release because the problems are likely to have been fixed. If not, please let us know.

It is now easier to move SAS data sets from one computer to another (from IBM to VAX, for example) because PROC COPY, which is used in transporting SAS data sets between operating systems, can now move, at one time, a whole SAS data library containing many SAS data sets instead of just one SAS data set at a time.

You will notice on the output of your SAS log that the content of the error messages has been improved somewhat. Instead of just receiving a generic error message saying "You made a syntax error," you will now get a much more diagnostic error message. In many cases it tells you exactly what is wrong and suggests ways to fix the error. This is an ongoing improvement. We have made this improvement in some areas, and in later releases you will see more and more of these better error messages to help you fix any errors you have made.

You also see CPU time messages after each DATA and PROC step. You can suppress these with the option NSTIMER, which is documented in the manual. (It stands for system STIMER through system timer.) On the minicomputers, you see not only the CPU time used by the staff, but various other statistics such as the number of page faults, the amount of I/O, and the elapsed time taken by the staff.

In Release 4.07, which will be shipped within the next few weeks to all people who have licensed the product, you will also notice that notes and error messages will be much more understandable because they will appear in the right place. In the earlier releases, you may have noticed that the note for one step appeared somewhere in the middle of the next step. Because we had several requests to fix this, it has been fixed. You will not see it in 4.06, but you will see it very soon in 4.07 when you receive that release.

There are a couple of other miscellaneous features that I wanted to mention. The \( x \) statement, the statement that sends commands to the host operating system, can now (under VMS) execute all VMS commands except LOGOFF, which we will not support.

The ones that you could not execute before were ASSIGN, SET, DEFAULT, and several others of that nature. Those now work. You do not have to leave your SAS session to do a LOGICAL; ASSIGN or to do many of the SET commands that previously did not work within the SAS System.

Another new feature that we have is the TLOG option, which was requested by several people. This option, which is available on one of the several IBM systems, is now available on the minicomputers as well. When you say OPTION TLOG, every SAS statement you type in during your session will be written automatically to a disk file that you can use later if you want to recall the information that you typed in.

The Display Manager already has a similar facility, but now you can use OPTION TLOG to do the same thing, even when you are not using the Display Manager.

And for some of the other progress that we have made, I will turn the discussion over to Steve Beatrous.

Steve Beatrous:

I would like to summarize some of the progress we have made in the DATA STEP. Bruce has mentioned some of the new features that we have implemented, but I think two of the most important things that we have done between Release 4.04 and the current release, 4.06, are fixing bugs and improving performance. I would like to highlight some of the specific bugs we have fixed in the DATA STEP and some of the specific things we have done to improve performance.

All INFILE and FILE options now work. In the earlier release, the HEADER= option, the EOF= option, the N= option, and the LINESIZE= option all had serious bugs in them, particularly when you had more than one FILE or INFILE. The OVERPRINT option was not implemented at all. Currently, all of these are fully implemented in 4.06.

Some functions in 4.06 returned bad numbers if the INPUT data were somewhat hazy; others would abend if you had an overflow or underflow. Some of the specific functions that had problems were the RANUNI, the SKEWNESS, and the TAN. All DATA STEP functions now work and will trap overflow and underflow errors correctly.

In previous releases, you could inadvertently rename a variable on a data set with a name that already existed on that SAS data set. This bug has now been fixed. You receive an appropriate error message at compile time if you try to rename a variable on a data set with a name that already exists.

Infinite loops of the system were very easy to get into in Release 4.04. If you interrupt your SAS session too early (for example, by pressing a CTYRY/L or ATTN key), or if you tried to access SAS in a directory to which you did not have the right access, the system went into an infinite loop, and the only way to break out of it was to disconnect your session. These have all been fixed, and it is, in fact, very difficult now to put the system in an infinite loop.
In several cases, there were abnormal terminations for things like missing an END or OTHERWISE statement in a SELECT group, or having more lines in the Display Manager than it could fit in its current memory space. Right now you receive clean, concise error messages before termination. In 4.04, you simply received a traceback and something very confusing. Now you get a message that says:

- in the case of running out of memory: You have run out of memory
- in the case of the SELECT group: You got a compile-time error message and syntaxing continues

These are the major debugging procedures that I wanted to highlight. In terms of performance improvement, a lot of the things I am going to highlight may sound esoteric, but let me start by saying that one of our benchmarking programs in 4.04 took 12 CPU minutes to run; in 4.06 it takes a little under 4 minutes to run.

We have significantly improved the performance of the DATA STEP between Releases 4.04 and 4.06 by implementing ways of doing almost all arithmetic operations directly in storage. In 4.04, we were doing almost all arithmetic operations by using a stack, pushing intermediate values on top of the stack, popping them off, adding them together, doing whatever manipulation was required, and then pushing the result back on top of the stack. In later releases we are planning to move away from stack-oriented manipulations entirely because that significantly slows down the run time of the DATA STEP. DO LOOPS have gotten significantly faster because we have placed the BY and TO values into registers rather than putting them on stacks. So, you may notice that DATA STEPS that did a lot of looping will now run much faster.

ARRAY access has been made about twice as fast at run time. In the SAS System, you currently have two types of arrays: what we call easy, or fast, arrays and what we call slow, or hard, arrays. You can have each element of an array represent a different length in the SAS System. For example, numeric values can range from length 2 to length 8. If all of the variables in your array have different attributes, then the variables in that array will be scattered all over memory and that would become a 'hard' array. All arrays used to be hard arrays, but most users will specify arrays of the 'easy' type where the variables are the same length. The run time support routines can process arrays a lot faster by assuming that they are all in contiguous storage, and we can access one element right after another.

There is an implicit iteration of every DATA STEP that goes through the number of observations of the number records in your input file. At the beginning of each iteration of the DATA STEP, we set every variable in the data set to missing. We were doing that on a variable by variable basis. We have now implemented a function that will essentially smear a storage area with missing values, so the process of iterating in a DO LOOP has gotten much faster.

Various heavily used PL/I routines have been rewritten in MACRO, including the routine that sets missing values and checks for missing values. Finally, I would like to touch on some improvements that we have made in EXTERNAL I/O; that is, reading and writing records defined by the FILE or INFILE statement and executed by the INPUT or PUT statements. Full-page report writing is now possible because the N= option is fully implemented. You can now write on the last line of a page and then go back up to the first line of the page which makes report writing a whole lot cleaner and neater. All of the SAS formats are now available. Some of the new formats include integer binary, packed decimal, and HEX. Formats are now available in PROCs as well as in the DATA STEP. The DFormat option on PROC FORMAT is now working, so it is possible to set up a library of permanent SAS formats that you can link to at a later point and keep around as a permanent record or a permanent set of formats.

That covers the items that I wanted to mention at this point. There are a lot of other things that we have done to improve the system, but I tried to pick out the ones that had the most relevance to you as a user and that were easier to describe to someone that does not know the code very well.

Richard Usanis:

Let's now move into what we plan for the near future. I will mention that as each of our speakers addresses you, the projects that are being discussed usually cover all parts of the system. Everyone will have a part in each one of the projects at some point, and what we try to do is orient the projects to the first level of concern about them. You need to keep in mind that everyone is involved in almost all of the projects. With that, I think that we will start with Mark Schaffer, who will handle the host interface and particularly the VAX-host interface, then we will move to Steve Beatrous, to Bruce Tindall, and finally, to Darrell Massengill to wrap up this session.

Mark Schaffer:

Most of the work that we will be doing in the next year will affect the performance of the SAS System under the VMS operating system. Currently, the whole system that you see is linked to one image so that when a user invokes the SAS System, he gets all of the procedures mapped into his user process, which consists of about 5 megabytes. In the future, all of the procedures will be dynamically loaded, enabling you to get most of the system (everything except
the procedures) in one base image, which will be about 1.5 megabytes. Each procedure will be linked to its own image, and when the procedure is invoked, it will be dynamically loaded into the user's process. This process will enable us to provide for the facility of user-written procedures under the SAS System for VMS only. This process will be similar to the one currently used on the IBM. You will be receiving a grammar processor (parser generator) and a document on how to call the supervisor routines.

We will be working on the I/O subsystem. In the past, the development of the I/O subsystem was done in response to development needs. They will all be bundled into one I/O subsystem and written in MACRO.

We will also be working on clustering the image, which means that the modules that call each other frequently will be linked close to each other, and this should not reduce the page faulting.

Steve Beatrous:

There are three things that I would like to highlight that we are going to be working on in the coming year. In this year's SASware Ballot, the item rated #1 was a stored program facility. This gives the user the ability to compile a SAS program consisting of a DATA STEP and a set of PROCs into some sort of executable image that could be invoked at a later point. A major benefit of such a system would be performance. Large production jobs would no longer have to go through the overhead of syntaxing a DATA STEP and a set of PROCs: they could be compiled the same way you would compile a PL/I or FORTRAN program into some sort of executable or object file.

Another benefit is security. Users who market their SAS software do not like to distribute the source of that software. If you do not have a stored program facility, the only way to market your SAS software is to distribute the SAS source statements. So, in implementing this facility, we will get a performance gain, and you will be able to market your SAS software more efficiently and effectively.

We planned the first design and implement facility to store just a single DATA STEP. We would like to store it in an executable file, be able to invoke it from another interactive SAS session, thus activating the run time for that particular DATA STEP. In the future, we plan to work on multi-step programs and on storing MACRO definitions. Bruce will be talking about what he plans to do for MACRO, so I will not go into that at this point.

One thing that we are very curious about with a stored program facility is exactly how it is going to interface with the SAS System. Currently we have some ideas, but I am particularly interested in your feedback. Would you like something like a RUN filename statement that would invoke an image? Would you like to invoke something as an executable file? I would very much appreciate it if people with ideas on this would bring their ideas to the Birds of the Feather session tonight or make a point of getting in touch with me after this talk.

Another thing we are planning to do in the current year is improve the performance of our DATA STEP interpreter. We are now working on two approaches to speed up the DATA STEP run time. One approach is to simplify the path that any given instruction would have to take through our interpreter. The other approach is to place absolute addresses into intermediate code.

Let me give you a little background on what this task might be so that what we are doing may become a little clearer. Currently our DATA STEP compiler emits an intermediate code which is interpretively executed by an interpreter. The intermediate code consists of Assembler-like instructions, but it is a pseudo-Assembler that does not exist on any machine unless you define our SAS System as a machine of sorts. Any given instruction in a DATA STEP can be executed virtually an infinite number of times. If we can speed up the path that any given instruction takes, we can greatly improve performance of the DATA STEP. We are looking at a lot of individual ideas, which are probably too esoteric to get into at this point, that would simplify the path that a given DATA STEP instruction would take. As I said before, this would have the benefit of improving the performance of your DATA STEP programs.

The other performance project that we have slated is placing absolute addresses into our immediate code. Right now, we place the address of a given variable into our intermediate code as a relative address. What we would like to do is have a preprocessor in our interpreter convert that relative address to an absolute address. In the current 4.06 release, when we are executing each instruction that has an address in it, such as an add instruction, we have to convert that relative address to its absolute before performing the addition. Actually, we already began with this task by the storage to storage instructions that I mentioned earlier in the progress reports, and, by making arithmetic operations go from a storage to storage location, we are placing absolute addresses into the intermediate code. That particular task improved the performance of a DATA STEP by about 25%. So, if we can make all instructions that contain addresses have absolute values rather than their relative addresses, then we can improve the performance of the DATA STEP significantly.

The last upcoming project that I would like to talk about is one related to memory management. One of the problems with the SAS System for minicomputers is its extravagant use of memory. This can be seen by the user through the amount of page faulting that is done by a SAS program. The task that we have planned for memory management should improve the performance of both the DATA STEP and many PROCs. The two memory problems that we currently have are:

- Large memory pools get fragmented. We may have a couple of megabytes of memory
available to the SAS System, but they are such small chunks that they are not in a useful format. What we plan to do to alleviate that problem is to try and come up with more logical points in a SAS job when we could do memory clean-up collapsing three pieces of storage into usable size chunks.

* The second problem that we are trying to address is currently on all of the operating systems under which we are running. We do not have a way of releasing large memory requests back to the operating system. This means that if you have a large DATA STEP or PROC in your SAS code, it grabs a huge chunk of memory, which will never be freed to the operating system. Your process is charged for whatever chunk you have allocated, but you are not using a very large portion of it. Essentially, your memory size grows to some maximum point and stays there; it never goes back down. So, the efficiency of DATA STEPs and PROCs that come later in your source is seriously degraded.

We are looking into ways of freeing very large memory requests back to the operating system so that your process will no longer be charged for large memory requests made by a PROC or a DATA STEP.

That pretty much covers the three things that I wanted to touch on at this point, so I will now turn the discussion over to Bruce Tindall.

Bruce Tindall:

I am going to talk little bit about what our plans are for the SAS macro facility for multidimensional arrays in the DATA STEP and for an intermediate code optimizer that will significantly improve the performance of the DATA STEP.

Several people have asked me about the SAS macro facility, and for those of you to whom I have not yet spoken, let me assure you that it is one of the projects that is going to be started immediately after we return from SUGI. It is one of the next things that will be available. We are going to implement it in the following stages. If you do not like the order in which we are going to do it, please let me know because this is completely open, within reason. We cannot implement the entire thing next week, but if you want certain features of it before certain other features—and some features are very important—please let me know.

The first feature we plan to implement is the %INCLUDE and %LIST facility. These are not, strictly speaking, part of macro, but everyone uses them in conjunction with macro, so those will be the first things to be implemented. You can already do some weird things using the SAS Display Manager; however, if you want to run in non-Display Manager mode, there is no way to do %INCLUDEs and %LISTs. Therefore, we will implement those first.

Next we plan to implement simple macros. These are simply macros that are straight substitution text, not including all of the fancy stuff like $DO, and %IF-THEN, and all of the programming statements in the new SAS MACRO facility. Those will be implemented later, and that will be the final phase of implementing SAS MACRO facility.

Later on in the production cycle we are planning to implement a completely new feature, multidimensional arrays. As the SAS System exists now, you can use two-dimensional arrays by making arrays of arrays on the IBM operating systems. However, it is extremely cumbersome to do. In the SAS System on minicomputers now, as we have noted before, you can have explicit indexing of arrays. This is available on the minicomputers, but you only have one-dimensional arrays. Presently, if you want to define an array that is a two-dimensional table of rows and columns or a three-dimensional table, or whatever you would call that, you cannot do it. However, the new feature will allow you to do that with explicit subscripts in which you do not have to figure out what your index variable is and where it is getting set, and so on. It will be right there in the same statement with the array reference.

One more performance improvement that is being worked on that we have not yet talked about is an optimizer for the intermediate code. As Steve mentioned, the SAS compiler emits intermediate code, which is then executed interpretively by the SAS interpreter. The first phase of this optimizer for the intermediate code, which uses a technique known in the literature as peephole optimization, will reduce the number of instructions that are executed in any DATA STEP by 40% to 50% in the best case and probably even 30% in the worst case. Of the instructions that remain, many of them will be faster than the ones they replace. So, you can see that this will have a very significant effect on the execution time of a DATA STEP, especially a DATA STEP that reads in large numbers of observations.

One kind of statement that will be speeded up dramatically by this first phase of the optimizer will be arithmetic expressions on which we have done some work, but they still have a long way to go. Complicated IF-THEN type tests, which are used very often for subsetting data, will be speeded up considerably by this optimizer. That is just the first phase of the intermediate code optimizer. There are more esoteric types that will be added later, all of which have bizarre sounding, but legitimate, names. The peephole is only the first part of it, but even when that is available, it will be a very dramatic improvement in the DATA STEP. This particular improvement will not affect the PROCs, but it will affect the execution of DATA STEPs.

One kind of statement that will be speeded up dramatically by this first phase of the optimizer will be arithmetic expressions on which we have done some work, but they still have a long way to go. Complicated IF-THEN type tests, which are used very often for subsetting data, will be speeded up considerably by this optimizer. That is just the first phase of the intermediate code optimizer. There are more esoteric types that will be added later, all of which have bizarre sounding, but legitimate, names. The peephole is only the first part of it, but even when that is available, it will be a very dramatic improvement in the DATA STEP. This particular improvement will not affect the PROCs, but it will affect the execution of DATA STEPs.

Now, to wrap things up, Darrell Massengill will tell you about some of the other plans we have.
Darrell Massengill:

I am going to talk about two short topics: SAS interface to non-SAS products and future machines. We currently have plans to interface to a number of non-SAS products. Our first level of implementation will be to database management systems. At the current time we have not chosen any particular products, and the products we choose will probably be different for each machine. But it is probably safe to say that we will interface to the vendor's database management package first, if he has one. If you have any particular preferences as to which database management package, or packages, you want interface to, please contact your SAS marketing representative and pass that information on to him.

We are currently investigating the next wave of machines on which to put the SAS System. As most of you probably know, the VAX/VMS version is now into production, the Data General AOS/VS version will soon be into production, and the Prime PRIMOS version is still under development but should be released in test form in the not-too-distant future. So, as a result, we are looking at the next level of machines that we are moving toward. Currently, we are looking at eight machines, but there have been no decisions as to which machine we will choose. I would guess that we would probably start to port the SAS System to two or three machines later this year. Since there are some machines that I cannot talk about and some I can, I will just not talk about any.

I will tell you a little bit about how we choose our machines, and that will give you an idea which machines we are looking at. Both the investigation and the porting process take a considerable amount of time. During the investigation process, we study the machine to determine whether it is possible to put the SAS System on the machine and how difficult a task it will be. We basically look at the following items:

- Does the machine have a PL/I subset G compiler?
- Does it have a virtual operating system?
- Is it a 32-bit machine?
- Does it have graphics capabilities?
- Does the machine have byte addressability; that is, byte addressability with PL/I pointers?
- Can we access system-level information?
- What are the limitations (program size, and so forth) of the machine?

Then there are a number of other special items that the SAS System needs to use with the machine that we must investigate.

Richard Usanis:

Thank you panel. I hope that this discussion was informative to all of you. If you do have questions, please bring them tonight. If you have ideas, please contact one or more of us today or tomorrow.

I think you will note that our emphasis is truly on improving the performance and making sure that the system is what you want it to be. If there is anything we can do to work toward that end, I think you will find us more than willing to do that.