PROTEUS / NEWS

AN INDEPENDENT NEWSLETTER FOR OWNERS AND USERS OF PROCESSOR TECHNOLOGY CORPORATION COMPUTERS

FORMERLY SOLUS NEWS

1983 PUBLISHED BY PROTEUS, 1690 WOODSIDE ROAD, SUITE 219, REDWOOD CITY, CA 94061, USA

VOL. 6 # 1
SINGLE ISSUE...$7.50 (US)
SINGLE ISSUE...$9.50 (FOREIGN)

Editorial

WHERE DO WE GO FROM HERE?

Every time I begin or end a new volume of Proteus News, I ask myself whether it is worth carrying on. As the years have passed, Proteus members have sold their Sols, lost interest, gone beyond the need for a user's group publication, or whatever, resulting in slowly diminishing membership. This is to be expected. The thirst for information was tremendous in the beginning of Sol's life, but by now the industry has matured and people have diversified in their interests and needs.

My original plan after Processor Tech liquidated was to keep the users' group active to disseminate the information we all needed or may have need, and I feel this has been done. The Encyclopedia Processor Technica compiled the manuals and other documentation into one reference collection. The content of Proteus News was restricted to items of interest to Sol/PTC hardware owners, since there was no other source.

I personally have learned a great deal about microcomputer hardware and software from you who have sent letters and articles for publication. But our members have been contributing fewer articles each year. It appears that the needs of our group have been met pretty well and I would like to free Proteus to expand to a broader scope. This doesn't mean that Proteus will cease to exist or that it will ignore the Sol. In fact, I really don't know what Proteus will evolve into in the coming years. But I want to allow it the freedom to go beyond a Processor Tech focus.

There is one important piece of Sol business that I would like to cover in this year. Since Sol was manufactured, the S-100 bus has become standardized by the IEEE society as the IEEE 696 S-100 bus standard. Processor Technology's use of the S-100 bus comes very close to the standard, but does not quite meet it in some respects. My goal for this year will be to examine the Sol and other Processor Technology hardware to document: (1) how it differs from the IEEE 696 standard, (2) how to alter it for full IEEE 696 compatibility, and (3) how to disable the 8080 microprocessor from the Sol's architecture to allow any IEEE 696 standard CPU board to take over the Sol, using the Sol as a slave device. This will let the Sol keep up with the rapidly changing technology of microcomputing. (See my article on future shock in this issue.)

I can't do this all myself, but I have looked into the topic already and plan to write enough to stimulate other

(continued on page 2, left column)

FUTURE SHOCK

Or

ARE 16 ADDRESS BITS ENOUGH?

by Stan Sokolow

We have all managed to survive quite nicely by ignoring the Z-80 microprocessor, but will we fare as well in the coming years? With the strength of IBM in the marketplace, the microcomputer industry is bound to leave the 8080 microprocessor family behind in favor of the 8088/8086. Some people feel the Motorola 68000 is a better processor, but IBM owns a chunk of Intel, so which microprocessor family do you think will come out the winner in the long run? I'll put my money on Intel's.

Intel has an impressive upward-compatibility plan. The 80186 puts a lot of the little support chips into the microprocessor. An article in Byte magazine (April 1983) compares the cost of an 8085 system, a similar 8086 system, and an 80186 system. The 80186 system cost was only slightly higher than the 8085, but it provides a 10-fold increase in computing speed (in MIPS, millions of instructions per second). The 8086 system cost about twice as much. (Only chip set costs were compared.) The 80186 provides on-chip: interrupt controller, two DMA channels, timers, clock generator, wait state generator, memory/peripheral chip select logic, etc. In fact, it is a CPU on a chip.

The 80286 is twice as fast (in MIPS) as the 80186. It is designed for multi-user/multi-tasking systems, having built-in memory management and memory protection, instruction pipelining (pre-fetch of instructions while prior instructions are still executing), etc.

Since we have just scratched the surface of potential demand for microcomputers, and everybody who's trying to be somebody is planning to ride on IBM's wake with an IBM-compatible computer, I'm sure that it won't take long before the number of 8080/Z-80 machines is only a small part of the installed computer base.

And what about the competition? Motorola's 68000 is making a good showing. (See Byte, April 1983.) National Semiconductor has an impressive architecture in their NSC 16032 family. (See Byte, April 1983.)

The software market will slowly wind down in the 8-bit machines, and all new things will come out for the 16-bit

(continued on page 2, right column)

(Editorial continued from page 1)

members to participate. If you are interested in following along with this discussion and learning more about the Sol and microcomputers in general, then I highly recommend that you obtain some reference material and background knowledge.

If you do not yet understand the basics of electronics (things like: voltage, current, resistors, transistors, diodes, MOS, etc.), then I recommend the book published by Radio Shack called "Understanding Solid-State Electronics". It is an excellent beginning and will carry you quite far.

For more information about TTL integrated circuits (like the multitude of 74LS... integrated circuits that are the building blocks of the Sol and other boards), I recommend Don Lancaster's "TTL Cookbook" published by the Howard W. Sams company, available through most electronics stores. You needn't master the whole book; the first few chapters will give you what you need.

A very practical book which I highly recommend is the paperback on the IEEE 696 bus written by Sol Libes and Mark Garetz, published by Osborne/McGraw-Hill, entitled "Interfacing to S-100/IEEE 696 Microcomputers". I have the 1981 edition, which was written when the standard was still in development. The final standard (approved at the end of 1982) differs slightly from the original proposal, so I expect that a revised edition of this book will be published. Mark Garetz, as chairman of the IEEE 696 standards committee, wrote an article in the February 1983 issue of Byte magazine, outlining the approved standard and pointing out how it differed from the proposed one. I recommend this article as a supplement to the book.

And of course you should have a copy of the Sol technical manual, giving the schematics and theory of operation. It was distributed with the early Sol computers, but as Processor Tech moved into the small business market they stopped giving it away with the computer. If you didn't get a big black notebook with your machine, you can get the same information in Volume 2 of the Encyclopedia Processor Technica, obtainable from Proteus.

I must admit that I have been growing tired after running the show for so long. My first issue of Proteus News, originally called Solus News, was in August of 1977. That's almost ancient history as microcomputing goes. But I feel revitalized by the decision to let Proteus expand beyond the Sol. I will continue to publish all of the Sol articles I can get, but we'll just slowly run out of steam if that's where we quit. I want to do more with new programming languages, high resolution graphics, virtual memory, artificial intelligence, and so on. These are going to require hardware beyond the capability of Sol's 8080. The Smalltalk-80 programming system from Xerox, for example, needs about 500K of virtual memory. With 256K memory chips becoming cost effective, we'll soon look upon the 64K address space of the 8080 as tiny.

Rather than discarding the Sol or disemboweling it by removing the Sol-PC, I think the most rational approach is to let the Sol become a generalized S-100 machine, keeping the I/O interfaces, keyboard, power supply, walnut sides, and all. Then we can install our chosen microprocessor CPU into a bus slot and update the Sol. Ideally, we should be able to flip a switch, disable the bus CPU, and re-enable the Sol's 8080 to run our old software whenever we want.

This project of documenting the conversion of Sol to a

(Continued at right, bottom)

(Future Shock continued from page 1)

systems with their larger memory space and extended instruction set. If we want to preserve the usefulness of our Sol, we should be considering how to update it to a new processor while still maintaining as much of the Sol as possible.

When I first bought my Sol, I bought a 32K static memory board for the introductory price of $1000. I can now buy 8 times that capacity on one dynamic S-100 board for half the cost. But my Sol's 16-bit address space can only use the extra capacity as a RAM-disk, since bank-select schemes are such a hodge-podge that no one really supports extended memory that way.

The S-100 bus provides us a way to make use of the cheaper hardware through extended addressing. But can we actually use this extra memory? What will we do with it all? And who needs the extra processor speed?

Well, applications always seem to have a way of growing to reach the capacity of the hardware. Visi-Calc people want bigger spread sheets and faster updates. Number crunchers want faster arithemetic with more significant digits. Word processor people want faster searches and multi-tasking. Programmers want smarter operating systems. I don't think we'll have any trouble using the larger systems as personal computers. In fact, I think future programming systems and non-procedural languages will require more address space than we now can support in the Sol.

Thanks to the IEEE 696 standard, the S-100 bus has gained some respectability -- it's not just a "hobby bus". Heath/Zenith is the first nationally recognized and established manufacturer to opt for the bus (in their Z-100 machine). I think the S-100 bus is here to stay for a while. Consequently, I am looking forward to upgrading my Sol someday.

Why bother upgrading the Sol instead of buying a new machine? Well, for one thing, I have one already. In fact, I have several which I use for business, and one to play with. I think there are great bargains to be found in old Sols being dumped at low prices. Also, I know the Sol well and am not afraid of modifying it. Its five slots are enough to build up a powerful system, especially when the Sol-PC is used for I/O. Its backplane has interleaved ground lines to minimize cross talk. I like the keyboard and have worked out some modifications that make it more versatile (to be described in another article). It is oriented for memory-mapped video which is more versatile than terminal-oriented video. And on, and on.

Besides, I like the walnut sides!

==========

(Editorial continued from left)

fully standard IEEE 696 machine is essential for keeping the Sol a contemporary computer in the coming decade. I hope you'll participate in the dialog and experimentation to achieve this goal.

Sincerely,

Stanley M. Sokolow, D.D.S.

SMALLTALK-80:

THE LANGUAGE FOR THE NEXT ERA?

by Stan Sokolow

When I first heard the name "Smalltalk", I thought it was a little language to teach kids about computers. You know, something like Tiny Basic. But I was wrong. Smalltalk is not small and it doesn't talk.

Let me quote from the book which is my source for this article:

> * Smalltalk is a vision.
> * Smalltalk is based on a small number of concepts, but defined by unusual terminology.
> * Smalltalk is a graphical, interactive programming environment.
> * Smalltalk is a big system.
>
> (Smalltalk-80: The Language and its Implementation,
> Adele Goldberg and David Robson.
> Addison-Wesley Publishing Company.
> 1983, page vii.)

Smalltalk is a language and operating environment developed by Xerox at their Palo Alto Research Center (PARC). People tell me it was inspired by some LISP-ish ideas, but it is also kind of FORTH-ish. I'm not enough of a programming language guru to comment on that. But I can say, from what I've read about it, it is a different way of looking at programming than I am used to from the contemporary languages (FORTRAN, BASIC, COBOL, PL/I, Pascal).

What's different about it? It has a different way of looking at computing. Instead of procedures which pass arguments, Smalltalk has "objects" which pass messages to each other. If the object understands the message, it will act upon it and send back a reply. For example, instead of adding 1 to a counter with an arithmetic assignment statement that acts upon variables, in Smalltalk you send the number (which is an object) a message to add 1 to itself. Since numbers know what addition is, the result is computed and returned as the reply.

This may sound absurd, but consider other possibilities. You could send the message "sort" to an array object or to a file object; or you could send the message "rotate" to an object which contains the representation of a molecule on your graphics display.

Now build up a library of definitions which tell the computer what classes of objects exist in your programming universe, and for each class what messages are understood by members of the class, and for each message what methods (here's where the programming comes in) are used to act upon the messages by members of the class.

Include in your definitions a bunch of classes of objects that programmers like to use: Sets, Bags, Dictionaries, OrderedCollections, SortedCollections, LinkedLists, Intervals, Arrays, Bitmaps, ByteArrays, Strings, Symbols, Numbers, Floating, Fractions, Integers, Files, Streams, FileDictionaries, and so on.

Add to your objects those items that are often thought of as parts of an operating system: ProcessorScheduler, SharedQueue, Processes, Semaphores. Let programmers send messages to these objects. For example, you could send the message "create" with some arguments to the ProcessorScheduler and create a new process (task) executing concurrently.

Then let the programmer build up methods which send messages to objects, get replies, pass replies on to other objects as messages, and so on. Let classes be defined as subclasses of others, so that they can inherit the methods of their parent classes without having to redefine them all over again.

What have you got? You are building a modular programming system that is extensible by the programmer and which allows him to think in terms that are appropriate to the problem he is solving. He can create new classes of objects, like "ComplexNumber", and define the operations (messages) it can act upon. Then in future programs, he can simply use the new class and its messages as though it were a part of the programming language all along.

If a better method for computing something is needed, just that class definition can be changed, and all other objects which use that class will use the new method without knowing it.

What have you got? You have the Smalltalk programming language and operating system.

But Smalltalk is not just a language and operating system. It is an interactive programming environment, too. Programmers can edit and debug their programs in their high-level language (Smalltalk). Running programs can be interrupted and viewed with a "browser" that will show the messages being processed, protocols which define the methods, etc. All of the programmer interaction is carried out at a high-resolution bitmapped display screen, with the aid of a cursor moving device (mouse), some buttons, and a keyboard. Most of the interaction is done by selection of items from menus using the mouse to point.

When an error occurs in the execution of a process, the process is suspended and the programmer is notified on the display. The error message (in words the programmer can understand) appears as the heading of a box on the screen, and in the box is a list of objects that are acting on messages and the messages the are processing. This is comparable to seeing a the stack of the procedure calls and arguments that were descended to get to the present place in a Pascal program.

The programmer can then get more information about the suspended state of the process, such as finding out exactly which statement was being executed and the values of all of the variables in the system. He can make changes to the source code of the "methods" (procedures) and resume the program where it left off.

All of this is done on the screen with overlapping boxes that look something like sheets of paper placed down upon each other. When the top one is removed, the next one down becomes visible again, until you finally get back to the original screen. The whole system itself is programmed in Smalltalk, except for the machine-coded interpreter.

Why is this interesting? There are a few reasons.

The interactive video ideas which came from Xerox PARC's work have been incorporated into some of the newest personal computing systems, such as the Apple Lisa and the VisiCorp product called Visi-on.

Intel has developed a new computer (called the iAPX 432) that is said to be an "object-oriented" machine. In other words, it is a machine which more closely implements the kind of architecture that is needed for the newest high-level languages.

Well if the Xerox PARC people like programming with objects, and Intel is making an object-oriented computer, I guess we ought to start learning a little more about objects. In fact, I suppose Xerox felt the same way, so they have announced the availability of commercial licenses and university licenses to the machine-independent Smalltalk-80 system.

Smalltalk-80 is the 1980 version of Smalltalk. It is implemented much like UCSD Pascal was; that is, in a pseudo-machine code which requires a machine-specific interpreter to execute. (It is NOT the same as the UCSD p-code interpreter.)

I'm sure we'll hear much more about Smalltalk in the future. If any Proteus member has access to a Smalltalk system, I would like to get some comments. Meanwhile, I recommend the book I mentioned, although you should be prepared to be confused by it the first time through. I have the three supplementary books on order, but they are still being written.

==========

THE IEEE 696 STANDARD AND SOL

Part 1: Compliance and Upgrading as a Bus Master

by Stan Sokolow

This is the first in a series of articles on the S-100 bus standard and how it affects Processor Technology products, with emphasis on the Sol computer. In this part, we will look at the standard and compare it with the Sol's use of the S-100 bus. We will show how to upgrade Sol to comply with the standard. Later, we will discuss how to bring other Processor Technology products into various degrees of compliance with the standard, and how to convert the Sol into a bus slave rather than a bus master so a new CPU board can be installed in the bus.

What we won't discuss to any great extent in this part is the timing relationships of signals on the Sol bus. The IEEE standard gives various timing specifications, but I have not investigated these in the Sol yet. If any members have found timing problems other than the few I mention, please let me know about them.

I want these articles to be a dialog among Sol owners so we can all pool our knowledge and experience. I don't pretend to be an expert, so I am counting upon you to collectively examine my proposals, find the flaws, make new proposals, comment on the correct ones, etc. After publication of each Proteus News issue, I hope to receive letters from you with corrections and improvements upon my proposals. I will organize the ideas and present the new information in the next issue.

In the series, I must assume a certain level of knowledge. If you are having a hard time following the technical jargon, please see my editorial in this issue for more background and references. The article in Byte, February 1983, by Mark Garetz is an excellent place to start.

INTRODUCTION TO THE IEEE 696/S-100 BUS STANDARD

The IEEE is the Institute of Electrical and Electronics Engineers. One function of the IEEE is the establishment of design standards, and the IEEE has been developing a written standard for the S-100 bus. This IEEE 696 standard has been adopted by the IEEE standards board on December 9, 1982. The standard describes the mechanical and electronic specifications for the computer bus commonly known as the S-100 bus.

The Sol computer uses an S-100 bus in the expansion card cage. Obviously, it was designed before the IEEE standard, but the standards committee carefully considered the existing S-100 hardware when proposing the standard. Consequently, most of the existing S-100 equipment (including Sol) comes very close to the adopted standard.

However, there are some differences. Some of the unused lines in the S-100 de facto bus have been defined to have functions in the standard. Some of the lines used by just a few manufacturers for special features have been defined to have new uses. Some of the lines which only had meaning for the 8080 microprocessor have been redefined so that the bus can accomodate a wide variety of microprocessors in an interchangeable manner. We'll examine these differences, signal by signal.

FUNCTIONAL ORGANIZATION OF THE IEEE 696 BUS

The IEEE 696 bus consists of 100 lines that carry the signals which communicate among devices in the computer system. Each line corresponds to a pin position in the 100 pin connector. If you hold an S-100 board in your hand with the component side facing you and the S-100 edge down, the pins will be numbered from left to right, 1 to 50, on the front (component) side. Flipping the board over so that the solder side is facing you and the edge connector is still down, the pins are numbered from right to left, 51 to 100, with pin 51 directly behind pin 1.

Most lines are only supposed to be in in one of two states (high or low, also know as +5 or 0 volts) except for brief moments when making the transition between these states. In reality, the standard specifies that any voltage less than or equal to +0.8 volts shall be recognized as low, and any voltage greater than or equal to +2.0 volts shall be recognized as high.

NOTATION

Each bus signal has a specific interpretation for the meaning of the high and low states. Those that are "on" (also known as asserted or active) when the line is low are given names that end with an asterisk and are said to be "active low" signals. Signals that are active when the line is in the high state have no asterisk. In electrical engineering notation and in Sol documentation, a bar over the symbol is usually used to designate active low, but the IEEE committee decided it was easier to type their documents on a word-processor if they used something instead of the bar, so the asterisk was adopted. For example, SIXTN*. In this series, I will also use the asterisk notation, even when referring to Sol signals that are shown in the Sol manual with a bar. Read the asterisk as "star", but think "bar".

BUS DRIVERS

The devices which generate the high and low voltages on the bus lines are known as bus drivers. There are three types.

Active drivers accept (sink) current in the low output state and provide (source) current in the high output state. If several active drivers are connected to the same bus line and one goes into the high state while another goes into the low state, a conflict occurs on the bus, which is not to be permitted. For this reason, the other types of drivers are used where several drivers must operate a bus line.

Open collector drivers sink current in the low output state, but they neither accept or provide current in the high output state. Open collectors allow several such drivers to be connected to the same line, which is pulled to a high state by a 1000 ohm resistor (the "pull-up resistor") to +5 volts somewhere in the system. When any of the open collector drivers go into the low state, the line is pulled low and the asserted driver sinks current through the pull-up resistor. Several can be asserted (low) at the same time. Others can be in the high state without creating a bus conflict.

Tri-state drivers are the third type. These devices have a disable input which can put the driver into a high-impedance state. In the high-impedance state, the driver is said to be disabled or floating, and it is virtually isolated from the bus line it drives, neither sourcing nor sinking current from the line. Some bus signals are used to disable the tri-state drivers on other bus lines, such as the address and data drivers from the CPU to the bus.

FUNCTIONAL GROUPING

The lines are functionally divided into 8 groups, although they are intermingled in their physical position without much apparent rhyme or reason. (Actually, the pin positions were determined by the guys who built the first S-100 computer, the Altair. It was simply a matter of layout convenience; they just ran the signals to the nearest bus pin.)

The functional groups are Address, Data, Status, Control Output, Control Input, Interrupt, TMA Control, and Utility.

MASTERS AND SLAVES

The devices that plug into the bus are divided into masters and slaves. Bus masters are devices that take control of the bus, such as microprocessor boards (CPU's). Bus slaves are devices that take orders from the bus masters, such as memory boards and input/output ports. In one system, there can be up to 16 temporary bus masters, one permanent bus master, and many bus slaves. Temporary bus masters are devices that can temporarily take control of the bus away from the permanent bus master in order to communicate directly with the slaves. Examples of temporary bus masters are direct-memory access (DMA) disk controllers or alterate CPU's.

The term "permanent bus master" is somewhat misleading because this master does yield control to the temporary bus masters, and therefore is not permanent. The major distinction is that the permanent master generates the system clock, and it is the primary master, having initial control and regaining control between temporary master accesses. (Primary and secondary would have been better terms than permanent and temporary, but I think the standards committee must have liked the sound of "TMA" as a generalization of the DMA concept.)

ADDRESS BUS

There are 24 lines in the address group. The 8-bit microprocessors, such as the 8080 in the Sol, only have 16 address pins since they can only address 2 to the 16th power (65,536) memory cells (bytes). Thus, the older 8-bit S-100 computers (like the Sol) only carried 16 address lines in their bus. The newer microprocessors can directly address more memory than that, so the bus has been extended by another 8 lines, giving 2 to the 24th power (16,777,216) directly addressable memory cells in the IEEE 696 bus.

The Intel 8086 microprocessor, for example, has 20 bits of effective address information in its architecture, so it can address about 1 megabyte of memory. The Motorola 68000 can address the full 16 megabyte space. At the moment, this sounds like a ridiculously large amount of memory, but as the cost of memory continues to drop, I'm sure that new programming systems and applications will make use of more memory than we can guess. Sixteen megabytes of address on the bus gives room to grow which should last for a few years.

The older 16-bit-address boards can still be handled on the bus if the total memory is restricted to 65,536 bytes (ignoring the 8 extended address bits) or if the boards are modified to recognize the 8 extended address lines. We will discuss extended addressing in more detail later.

The addresses on the bus always refer to bytes, even if the device holds 16 bits in each physical memory cell. That is, in 16-bit memory boards, the low order address bit will refer to the left or right byte of the 16-bit word at each cell. The other 15 bits of the address will refer to the actual cell number. The byte accessed with the low order address bit set to 0 is referred to as the even-addressed byte (or the even-data byte), and correspondingly the other byte is refered to as the odd-addressed byte (or odd-data byte) because its address ends in a 1.

DATA BUS

The IEEE 696 bus also has extended the data line group. This was not done by assigning any more data lines, but rather by changing the data bus group into a bidirectional one. The original S-100 bus had separate lines for data flowing to the microprocessor and data flowing from the microprocessor. These were called the Data In and the Data Out lines (DI and DO). Each consisted of 8 separate lines which carried 8 bits of data only one way (to the CPU or away from the CPU). In the IEEE 696 bus, the 16 data lines can carry data to or from the microprocessor, with lines in the control bus acting as the traffic signals. This has effectively doubled the capacity of the data bus.

The designers of the Sol actually used a bidirectional data bus anyway. They have described their reasons for doing this in one of the early articles on Sol that is reprinted in the Volume 1 of the Encyclopedia Processor Technica. So, as you notice in your Sol schematics, the Sol's S-100 data lines are referred to as the DIO lines (data in/out). But still the Sol only has an 8-bit processor, so both halves of the DIO bus carry identical bytes of data. For example, on output the Sol sends a byte of data to memory on the 8 DO lines and also on the 8 DI lines. The DI and DO lines always carry identical information in the Sol.

The IEEE standard actually allows for a mixture of 8-bit and 16-bit data devices in the same system. A new pair of

signals (sixteen-request and sixteen-acknowledge) allow bus masters to ask for 16-bit transfers if desired, and to discover from the addressed slave device whether it will put 8 bits or 16 bits on the data bus lines. Older devices, like our 8-bit memory boards, will not know about the new sixteen request, so they will not acknowledge, and the master will know to look for only 8 bits of data on the appropriate side of the data bus. This is a clever way to let us make use of our old equipment while extending the system to 16 bit processors.

The first proposal for extending the data bus named the new 16-bit path the DATA lines (rather than DI and DO) and numbered them from 0 to 15, with 0 being the lower order bit. But the adopted standard has changed this interpretation of the data bus. Since some microprocessors put 16 bit data out with the high-order byte first and some with the high-order byte second, there arose a conflict when trying to make the bus independent of microprocessor architecture. Did bus lines DATA 8 through DATA15 contain the high-order byte or the low-order byte?

The standards committee finally solved the dilemma by designating one side of the data bus for the even-addressed byte and the other for the odd-addressed byte. That way, the data is always consistent on the bus and it is up to the CPU to look at the proper part of the bus when interpreting the data. Some CPU boards will have to flip the data around for their microprocessor and others won't.

The data bus now has two naming conventions. In 16-bit transfers, there are the ED and OD lines, each numbered from 0 to 7. ED lines carry the even-addressed byte of data. In 8-bit transfers, the ED lines are designated DO, since they use the same lines as the data-out lines in the pre-standard S-100 notation. Likewise the OD lines are the same as the DI lines. That is, in 16-bit transfers of data, the odd-addressed byte is put onto the Data-In lines and the even-addressed byte is put on the Data-Out lines, regardless of the direction of transfer. In 8-bit transfers, the old uni-directional path (DI and DO) still is used so old boards will meet the standard.

Confusing? Yes, but we can probably ignore the mess unless we want to put a mixture of 16-bit microprocessors on the same S-100 bus.

STATUS BUS

The status bus group is used by any device in the system to determine the kind of bus cycle the current master is in. A slave device looks at the status lines to determine whether it should respond to the address and data signals. The status lines have names beginning with a lower case "s". Almost all are found in the Sol, except the new sXTRQ* sixteen-request line. The 8 status lines will be covered in detail in the numeric rundown of the bus pins.

CONTROL OUTPUT BUS

The control output bus gives 5 timing signals known as strobes. The names of these signals all begin with the lower case letter "p" because they originate on the current master (processor) board. They will be covered in the numeric section.

CONTROL INPUT BUS

Six lines make up the control input bus. They send signals from slaves to the master to tell the master when to do something special. The RDY and XRDY tell the master to wait longer for the data to be valid during bus transfers from the slave to the master. The HOLD* signal requests that the permanent master stop and relinquish control of the bus to another master. The INT* signal is the interrupt signal to the master which can be disabled (masked) under software control. The NMI* signal is a non-maskable interrupt request generally reserved for interrupts that must occur no matter what, such as the interrupt to process the power-failure detection condition. And SIXTN* tells the current master that the slave can send or receive 16-bit data as requested.

VECTORED INTERRUPT BUS

The devices which can interrupt the processor can be ranked into 8 priority levels, each sending an interrupt request on a separate line. Together these make the vectored interrupt bus. Somewhere in the system these lines must be connected as inputs to an interrupt controller that decides when to signal the CPU on the INT* line.

The Sol doesn't contain a vectored interrupt controller to arbitrate among these requests, but the old S-100 bus definition did include the lines because other S-100 manufacturers did make interrupt controller boards. If interrupts are to be used in a Sol, the Sol's address decoder circuit must be modified by jumpers as described in the numeric section, and an interrupt controller should be added.

TMA CONTROL BUS

The concept of "Direct Memory Access" (DMA) has been extended to cover access to the bus by any of 16 temporary bus masters. The process of transferring control from the permanent master to a temporary master is now called "Temporary Master Access" (TMA). The 16 temporary masters use 4 new lines to signal their priority level to the other masters and to decide which one of them is to gain control if several request access at the same time. This is known as bus arbitration. Another four lines are used by the temporary masters to disable the permanent master's bus drivers, allowing the temporary master to control the bus.

TMA access is too complex for me to explain in this series, but the book by Mark Garetz does it well. The lines used will be mentioned in the numeric section of this article. In a later part, I'll look at the way the Helios controller gains access to the bus to see how well it complies with the IEEE 696 standard.

UTILITY BUS

The other lines on the bus have been lumped into this miscellaneous category. It includes the power and ground, clocks, reset, clear, and error lines. These are explained in the numeric section.

Three lines have been left explicitly undefined (called the NDEF lines), so that manufacturers can tuck their own unique signals somewhere. The NDEF lines must be clearly explained in the manufacturer's documentation and it is up to the user to avoid conflicts when integrating various hardware using the NDEF lines.

Four lines have been reserved for future use by the standard and should not be used for any purpose. These are designated RFU, Reserved for Future Use. Sol puts 8080 processor status signals onto two of the RFU lines, but they are easily removed as will be explained in the numeric section.

COMPLIANCE OF THE SOL BUS, PIN BY PIN, (NUMERIC SECTION)

This is a pin-by-pin comparison of Processor Technology Corporation's use of the S-100 bus with the IEEE 696 standard adopted December 9, 1982.

****
Please note: These updates are still in the proposal stage. Do not make the changes in your Sol, except on a test basis.
****

Pin 1: Standard signal +8 v

This is the main power supply. The standard calls for no less than 7 volts instantaneous minimum, no more than 25 volts instantaneous maximum, and average less than 11 volts.

Compliance: Although the Sol's specifications comply with the standard, some Sol's were made with an improperly specified transformer in the power supply. Processor Tech issued an update calling for these Sols to be outfitted with an additional transformer (called a bucking transformer) to lower the secondary voltage. Check your Sol's backplane voltage with a DC setting on the voltmeter, pin 1 to pin 50 (ground), being careful not to short circuit any other pins. This will give you the average voltage on the line. To measure instantaneous voltages, an oscilloscope is required.

Upgrading: Methods for lowering excessive backplane voltage have been described in back issues of Proteus (Solus) News. The bucking transformer update is in the Sol technical manual.

Pin 2: Standard signal +16 v

The standard calls for 14.5 v to 35 v instantaneous, and average voltage less than 21.5 volts.

Compliance: The Sol specifications are within the range. Check your Sol if in doubt. !BE VERY CAREFUL NOT TO SHORT CIRCUIT YOUR POWER SUPPLY BY ACCIDENTALLY CONTACTING BUS PIN 2 (+16 V) AND BUS PIN 52 (-16 V) SIMULTANEOUSLY WITH YOUR VOLTAGE PROBE! They are exactly opposite each other in the bus connectors. An in-line fuse would be a wise safety precaution to add between the Sol power supply and the S-100 backplane power connector.

Pin 3: Standard signal XRDY

External ready signal, used by an external device to signal the master that the bus is not ready. The bus is ready when both XRDY and PRDY (pin 72) are asserted; when either one is low, the master will wait.

Compliance: Sol complies. Sol only generates wait states by pulling the PRDY line low, not the XRDY, but the Sol does respond to both XRDY and PRDY from the bus when generating the READY signal to the Sol's 8080 microprocessor.

Pins 4 thru 11: Standard signals VI0* thru VI7*

Vectored interrupt request lines.

Compliance: Sol doesn't use the lines, so Sol is in compliance by default. Vectored interrupt signals have no effect on Sol unless an interrupt controller is added and the sINTA line (pin 96) is modified as described below.

Pin 12: Standard signal NMI*

Non-maskable interrupt.

Compliance: Sol designated it the XRDY2 line, but Sol doesn't use it, so Sol complies as a non-implemented feature.

Pin 13: Standard signal PWRFAIL*

The power-failure detection signal. The line goes low at least 16 ms before the power supply lines will drop so low that voltage regulators will fail to regulate. It is not specified as an open collector line, but it is required to meet the specifications for an open collector (see decription in driver section).

Compliance: Sol doesn't use the line, so Sol complies as an unimplemented feature, but a pull-up resistor should be added since this line acts like an open-collector. See the section below on open-collector lines.

Pin 14: Standard signal TMA3*

One of the TMA arbitration lines. These are used by temporary bus masters to communicate bus-request priorities among each other.

Compliance: Unused by Sol, so Sol itself complies. However, the Helios (which will be examined in a future article) is a non-complying TMA device unless modified. It doesn't use this type of arbitration.

Pin 15: Standard Signal A18

Extended address line 18.
Compliance: Unused in Sol. See extended addressing.

Pin 16: Standard Signal A16

Extended address line 16.
Compliance: Unused in Sol.

Pin 17: Standard Signal A17

Extended address line 17.
Compliance: Unused in Sol.

Pin 18: Standard Signal SDSB*

Status disable; disables the processor's bus drivers on the 8 status lines: sMEMR (pin 47), sM1 (pin 44), sOUT (pin 45), sINP (pin 46), sWO* (pin 97), sINTA (pin 96), sHLTA (pin 48), and sXTRQ* (pin 58).

Compliance: The Sol also disables 8 status lines, but they are not quite the same. The Sol has sMEMR, sM1, sOUT, sINP, sWO*, sINTA, and sHLTA; but sXTRQ* is not implemented in the Sol because it is an 8-bit processor. See pin 58 below. Instead, the eighth status line in the Sol is pin 98 SSTACK, which has been redefined in the standard. See pin 98 below.

Upgrade: Make the pin 98 and pin 58 upgrades. Since Sol-PC does not generate sXTRQ* (pin 58), but other devices placed in the bus may respond to it, sXTRQ* must be held high. This is done in the pin 58 upgrade by using the driver made available after pin 98 is upgraded. This driver is disabled by SDSB*, so it will now meet the standard.

Pin 19: Standard Signal CDSB*

Control-output disable: disables the 5 control output lines: pSYNC (pin 76), pSTVAL* (pin 25), pDBIN (pin 78), pWR (pin 77), pHLDA (pin 26).

Compliance: This signal is called C/C DSB (command/control disable) in Sol documentation. Sol disables 6 control lines instead of 5. The extra line in Sol is the PINTE pin 28 line which has been dropped from the standard and instead is an RFU line; see pin 28. Sol also disables a control output signal PWAIT (pin 27) which has been re-defined as an RFU line in the standard. And the phase 1 clock which is used for standard signal pSTVAL* (pin 25) is not disabled by C/C

DSB. So, Sol needs upgrading.
Upgrade: See the upgrades for bus pins 25, 26, 27.

Pin 20: Standard Signal 0 v (ground)

A grounded line.
Compliance: Sol doesn't use this pin. In the Altair and Imsai it was used for the UNPROTect signal.
Upgrade: Jump pin 20 of the bus to ground pin 100.

Pin 21: Standard Signal NDEF

Not to be defined in the standard. This is reserved for machine specific use.
Compliance: Not used in the Sol.

Pin 22: Standard Signal ADSB*

Address disable. Disables all 24 address line drivers of the permanent bus master (CPU).
Compliance: Sol disables the 16 address lines, but not the 8 extended address lines. However, Sol does not drive the extended address lines, so it lets them float and thus complies. See extended addressing section.

Pin 23: Standard Signal DODSB*

Data output disable. Disables the DO7 through DO0 lines in 8 bit transfers. Disables all OD and ED lines in 16 bit transfers.
Compliance: Sol disables the 8 DIO lines. Since the DI and DO lines are tied together in Sol, all 16 lines are disabled by the DODSB* signal. So, Sol complies. However, when the lines are driven in 16 bit transfers, there will be conflicts between the lines tied together.
Upgrade: See the section on the 16-bit data lines.

Pin 24: Standard Signal Ø

Phi, the master timing signal for the bus.
Compliance: In the Sol, the timing signals are those needed for the 8080 microprocessor clocks: Ø1 and Ø2 (phase 1 and phase 2). The Sol uses pin 24 for the phase 2 clock, which meets the requirements of the standard signal Phi. So, Sol complies.

Pin 25: Standard Signal pSTVAL*

Processor status valid strobe.
Compliance: The Sol and other 8080 based S-100 systems use this for the phase 1 clock, which also meets the requirements of the signal pSTVAL*. The standard requires that pSTVAL* be gated by pSYNC to be used. Sol doesn't use this signal for anything other than phase 1, but generates it for the bus, so Sol complies. However, the Sol doesn't disable this signal when CDSB* pin 19 is asserted (low), so in this respect, Sol doesn't fully comply. If a temporary bus master tries to take over the bus and drives the signal pSTVAL*, a bus conflict will occur.
Upgrade: Abandon the driver at pins 4 & 5 of U77 (which is not disabled by CDSB*), and instead use the driver at pins 2 & 3 of U50 (which is disabled by CDSB*.) This U50 driver was abandoned by the pin 27 upgrade. To do this, cut the traces at pins 4 & 5 of U77 and install a jumper from pin 6 of U92 to pin 2 of U50 and one from pin 3 of U50 to bus pin 25. You must also swap the IC's U50 and U77 so that the pin 25 driver remains an 8T97, since this is required by the Sol design. The driver at pins 4 & 5 of U77 will be used by the SLAVE CLR* pin 54 upgrade.

Pin 26: Standard Signal pHLDA

Hold acknowledge.
Compliance: Sol generates pHLDA, so it complies.

Pin 27: Standard Signal RFU

Reserved for future use by the standard.
Compliance: Sol puts the 8080 WAIT status onto this line, so Sol doesn't comply.
Upgrade: Since Sol doesn't need the WAIT status, cut the trace from pin 3 of U50 to the bus pin 27 and from the 8080 pin 24 to pin 2 of U50. This will remove Sol from this line and abandon the driver, which is to be re-used for another purpose. (See the Helios article in subsequent part of this series for corresponding modification to the Helios controller.)

Pin 28: Standard Signal RFU

Reserved for future use by the standard.
Compliance: Sol uses this for the 8080 signal PINTE, the interrupt enable line. This is not available from most other microprocessors, so the standard has dropped the signal.
Upgrade: Simply cut the trace leading to pin 28 of the Sol S-100 bus.

Pins 29 through 34: Standard signals A5, A4, A3, A15, A12, A9

Address lines.
Compliance: Sol complies.

Pins 35 & 36: Standard signals DO1/ED1 and DO0/ED0

Data out / even data lines.
Compliance: See the section on the data lines.

Pin 37: Standard signal A10

Address line.
Compliance: Sol complies.

Pins 38 through 43: Standard data lines (various)

Data in/out odd/even lines.
Compliance: See the section on the data lines.

Pin 44: Standard signal sM1

The status signal that indicates current cycle is an op-code fetch.
Compliance: Sol complies.

Pin 45: Standard signal sOUT

The status signal indicating bus transfer to an output device.
Compliance: Sol complies.

Pin 46: Standard signal sINP

The status signal indicating bus transfer from an input device.
Compliance: Sol complies.

Pin 47: Standard signal sMEMR

The status signal indicating transfer from memory to a bus master other than interrupt-acknowledge or op-code fetch cycles.

Compliance: Sol complies.

Pin 48: Standard signal sHLTA

The status signal that acknowledges that a halt instruction has been executed.
Compliance: Sol complies.

Pin 49: Standard signal CLOCK

A 2 MHz clock signal (plus or minus 0.5% tolerance with a 40-60% duty cycle) which is not required to be synchronous with any other bus signal, but is available for use by such devices as baud-rate generators or counter-timers.

Compliance: Sol drives CLOCK with the inverse of the signal which can be found at point B of the clock circuit option jumpers (same as pin 10 of U90). The timing diagrams in the Sol manual show that this signal has a period which is either 5, 6, or 7 times the dot clock period of 70 ns, depending upon the clock jumpers. With the usual jumpering, the Sol generates a 2.045 MHz clock, which comes close to the standard for pin 49 CLOCK but is out of the 0.5% tolerance range. At higher clock frequencies, the Sol is even farther from the standard for pin 49.

Upgrade: For most purposes, 2.045 MHz is close enough, so no upgrade is needed if Sol is kept at the usual 2.045 MHz frequency. However, some systems may use this signal to clock a timer that keeps track of real time, so accuracy may be very important. If you want strict compliance with the standard, or if you have altered your Sol to have a faster clock cycle, you will need to find an alternate way to generate the 2 MHz clock.

There are hybrid circuits available now which generate a TTL clock in one small package the size and format of a 14-pin DIP package. Jameco Electronics (1355 Shoreway Road, Belmont, CA 94002, 415-592-8097) sells a 2.0000 MHz TTL oscillator for $9.95 in their 1983 catalog (page 21). It only requires connecting power (+5v), ground, and the output line. You can install it in one of the spare spaces in the Sol-PC, either U103 or U82. Since those are 16-pin DIP spaces, you'll have to align the oscillator at one end and jumper the oscillator's power or ground pin to the correct hole at the other end.

Then cut the trace at pin 14 of U77 which is the input to the CLOCK bus driver and jumper the output of the hybrid oscillator to that pin of U77. Be careful not to make the cut along the trace from U45 to U109. I haven't examined the Sol-PC to see where the traces are accessible, so you'll have to study this upgrade on the Sol CPU schematic and on the board.

Pin 50: Standard signal 0 V (ground)

Signal ground, common with pin 100.
Compliance: Sol complies.

Pin 51: Standard Signal +8 V

The 8 volt power supply, common with pin 1.
Compliance: Sol complies.

Pin 52: Standard signal -16 V

Instantaneous maximum less than -14.5 volts, instantaneous minimum greater than -35 volts, average minimum greater than -21.5 volts.
Compliance: Sol specifications comply.

Pin 53: Standard Signal 0 V (ground)

Common with pin 100.
Compliance: Sol doesn't use this line, although documentation refers to it as the SSWI sense switch input. (It must have been used in the Altair and Imsai front panels.)
Upgrade: Jumper pin 53 to pin 100 ground.

Pin 54: Standard Signal SLAVE CLR*

A reset signal which resets only the bus slaves. Must be active when POC* is active, but may also be asserted by external means (such as a slave reset button).

Compliance: Sol has no connection on pin 54, so it fails to assert SLAVE CLR* when it asserts POC*. (Most slaves don't care anyway, however.)

Upgrade: Use the abandoned gate at pins 4 & 5 of U77 to gate POC* onto SLAVE CLR* also. That is, make the pin 25 pSTVAL* upgrade which cuts the traces to and from these pins of U77. Then install a jumper from pin 12 of U77 to pin 4 of U77 and a jumper from pin 5 of U77 to bus pin 54. Whenever POC* is asserted on the bus, SLAVE CLR* will also be asserted.

Pins 55, 56, & 57: Standard signals TMA0*, TMA1*, TMA2*

Three of the TMA arbitration lines.

Compliance: The Sol does not connect to pin 55 or 56, but it does make non-standard use of pin 57 for a signal called DIG1* (Data Input Gate 1). This is used by the ParaSol to force PDBIN low in the Sol being tested. Since it is machine specific, it should be relocated to an NDEF line.

Upgrade: Cut the trace at S-100 pin 57, jumper that trace to pin 66, and change the Sol documentation. A similar change should be made on the ParaSol debugger board.

Pin 58: Standard Signal sXTRQ*

Sixteen request, the signal which the master uses to request a 16-bit data transfer.

Compliance: Sol uses this as FRDY*, which disables MWRITE drivers when a front-panel is controlling the bus, as in the Altair and IMSAI. The standard provides no way to disable MWRITE (standard signal MWRT, pin 68), so FRDY* is not needed to meet the standard. However, the ParaSol debugger uses FRDY* to control the Sol being tested. So FRDY* should be relocated to an NDEF line. Also, pin 58 must be driven into the high state to assure that it will not be asserted by floating to a lower level. (It is not an open-collector line.)

Upgrade: The trace from bus pin 58 to pin 1 of U49 should be cut and then pin 1 of U49 should be jumpered to pin 65 of the bus, which is specified as NDEF. A similar change must be made on the ParaSol board to relocate FRDY*. Also, cut the trace leading to pin 2 of U107 and jumper pin 2 of U107 to +5 (pin 16 of U107). Then cut the trace leading to pin 3 of U107 and jump it to sXTRQ* pin 58 of the S-100 connector. This will keep sXTRQ* high but allow it to float when disabled by SDSB*.

Pin 59: Standard Signal A19

One of the extended address lines.
Compliance: No connection in the Sol. See extended addressing section.

Pin 60: Standard Signal SIXTN*

The sixteen-acknowledge signal generated by bus slaves in response to the sXTRQ* request signal.

Compliance: Unused in the Sol. Should have a pull-up resistor added to indicate an 8-bit transfer and to meet the requirement for pull-ups on open collector lines. Jumper a 1000 ohm resistor from pin 60 to +5 volts.

Pins 61 through 64: Standard signals A20 through A23

Extended address lines.

Compliance: No connection in the Sol. See the section on extended addressing.

Pin 65: Standard Signal NDEF

Not to be defined in the standard, so available for machine specific use.
Compliance: Sol does not use this line, but it is needed to carry the signal FRDY* which must be relocated from pin 58 of the Sol bus. See the pin 58 upgrade and re-label this line as FRDY* in the Sol documentation. Make corresponding changes to the ParaSol debugger board.

Pin 66: Standard Signal NDEF

Not to be defined in the standard, so available for machine specific use.
Compliance: No connection in the Sol, but needed to carry the Sol-specific signal DIG1* which must be relocated from pin 57.
Upgrade: See pin 57 upgrade.

Pin 67: Standard Signal PHANTOM*

The phantom device disable signal. When PHANTOM* is asserted (low), normal slave devices will disable themselves and phantom slave devices will enable. Primarily used to disable RAM which is addressed where a ROM needs to be during initial instruction fetch after RESET* or POC*.
Compliance: The Sol generates PHANTOM* for the first four clock cycles after RESET* or POC*. During this time, the 8080 fetches its first instruction from addresses 0 through 3, which in actuality are located in the Sol ROM. PHANTOM* also is used to force the Sol's address decoder to respond to the address bus as though the Sol's ROM were addressed at 0. Therefore, Sol complies as a master which generate PHANTOM* and as a phantom slave. However, if any device on the bus tries to assert PHANTOM*, there will be a conflict with the Sol driver since it is a permanently enabled tri-state driver that is normally in the high output state, rather than an open collector driver.
Upgrade: Install a 1000 ohm pull-up resistor from pin 67 to +5 volts, and put a germanium diode in place of the phantom jumper on Sol, cathode (banded end) toward U77. When U77 pin 9 is low, current will flow from the pull-up resistor, through the diode, and into U77. The voltage drop across a germanium diode is small, about 0.25 volts, so pin 67 will be close enough to 0 volts to be recognized as the low state. When U77 pin 9 is high, the diode prevents current from sourcing back to the bus. Thus U77 acts like an open collector, sinking current when low, but not sourcing current when high.

Pin 68: Standard Signal MWRT

Memory write. Generated from two bus signals by the logic equation: MWRT = pWR and not sOUT.
Compliance: Sol generates MWRT just as described. In addition, although the standard does not provide an MWRT disable signal, the Sol uses the non-standard FRDY* signal to disable the MWRT driver. See pin 58 upgrade.

Pin 69: Standard Signal RFU

Reserved for future use by the standard.
Compliance: No connection in the Sol.

Pin 70: Standard Signal 0 V (ground)

Common with pin 100.
Compliance: Already grounded in the Sol.

Pin 71: Standard Signal RFU

Reserved for future use.
Compliance: No connection in the Sol.

Pin 72: Standard Signal RDY

The processor ready signal which when low causes the CPU to wait for a slow slave.
Compliance: Sol complies.

pin 73: Standard Signal INT*

The maskable interrupt request signal.
Compliance: Sol complies.

pin 74: Standard Signal HOLD*

The hold-request line used during TMA.
Compliance: Sol complies.

pin 75: Standard Signal RESET*

Resets bus masters; must be asserted when POC* is asserted.
Compliance: Sol complies.

Pin 76: Standard Signal pSYNC

The control signal identifying the initial clock cycle in a bus cycle.
Compliance: Sol complies.

Pin 77: Standard Signal pWR*

Identifies the presence of valid data on data bus.
Compliance: Sol complies. See the data bus section.

Pin 78: Standard Signal pDBIN

Requests data transfer from slave to master using the data bus.
Compliance: Sol complies. See the data bus section.

Pins 79 through 87: Standard Signal Address lines (various)

Address lines in the primary 16-line address bus.
Compliance: Sol complies.

Pins 88 through 95: Standard Signal Data bus lines (various)

Bidirectional data lines.
Compliance: Sol complies. See data bus section.

Pin 96: Standard signal sINTA

Interrupt acknowledge.
Compliance: Sol generates the sINTA signal, but the address decoder does not properly respond to interrupt acknowledge states unless the Sol is modified.
Upgrade: Revision E Sol's have two jumpers that should be installed; connect AE to AC and connect AB to AD. Revision D Sol's do not have these jumpers and the IC's are used somewhat differently. The following changes should be made to a Rev D Sol-PC: Jumper pin 8 of U57 to pin 6 of U34; jumper bus pin 96 to pin 9 of U57. This uses a spare gate in the Rev D Sol.

Pin 97: Standard Signal sWO*

Identifies a data transfer from master to slave.

Compliance: Sol complies. However, you should be sure to make the change described by Processor Technology in their Sol Rev N to Rev P change notice. This change corrects a potential data bus timing conflict during read cycles. I'm not sure when the conflict arises, but the change apparently alters the generation of the data bus driver disable signal OUT DSBL* found at the U47 output pin 6 (below the 8080 on the Sol CPU schematic). Instead of using the internal signal DBIN* which comes from the processor and not the S-100 bus, the modification will use the bus signal sWO* to determine when to disable the drivers.

The change consists of disconnecting the output of U45 pin 10 from the input of U47 pin 5 and jumpering the bus pin 97 (sWO*) to the input pin 5 of U47. This change will be essential if the Sol is to be made a slave of another CPU on the bus, since this disable signal would not be activated from the bus without it.

Pin 98: Standard Signal ERROR*

Signals a bus error condition during the present bus cycle, such as a parity error detected by a memory device.

Compliance: Sol puts the 8080 status SSTACK onto this pin, but it is not needed and should be removed to avoid asserting ERROR*.

Upgrade: Cut the trace leading to pin 98 if the bus. Install pull-up resistor as mentioned in section below on open-collector lines.

Pin 99: Standard Signal POC*

Power-on clear signal for all bus devices. When this signal goes low, as during a power failure, it must stay low for at least 10 ms.

Compliance: Sol generates the POC*. I'm not sure about the 10 ms recovery time, but it should be possible to calculate it from the data on the schematic.

Pin 100: Standard Signal 0 V (ground)

Ground.
Compliance: Sol complies.

OPEN COLLECTOR PULL-UP RESISTORS

The Sol generally uses 1500 ohm pull-up resistors, but the standard calls for 1000 ohm resistors on open collector bus lines. This probably will never make any problems. However for full compliance, the following resistors should be changed to 1000 ohm 1/4 watt resistors (5% tolerance):

R41 (SDSB*)
R20 (CDSB*)
R36 (ADSB*)
R33 (DODSB*)
R34 (RDY)
R31 (INT*)
R56 (HOLD*)
R55 (RESET*).

The open collector bus lines which need to have pull-ups added are:

pins 4 thru 12 (the interrupt lines),
pin 13 (PWRFAIL* acts as pseudo-open collector)
pins 14, 55, 56, and 57 (the four TMA lines),
pins 54 (SLAVE CLR*),
pin 60 (SIXTN*),
pin 67 (PHANTOM*),
pin 98 (ERROR*).

That is, connect a 1000 ohm resistor from these bus pins to a +5 volt source. Since the power to the bus is only the unregulated supply, the pull-ups will have to be connected to the +5 volt source for the Sol-PC itself. See the article on Bob Marsh's backplane board in this issue.

SUMMARY OF PART 1

The S-100 bus in the Sol requires changes to the following bus pins to bring them up to the IEEE standard 696 for a permanent bus master:

1 (nominal +8 volt line is overvoltage in some Sols)
4-12 (vectored interrupt lines need pull-ups)
13 (PWRFAIL* needs pullup)
14 (needs pullup)
20 (ground)
25 (pSTVAL* should disable by CDSB*)
27 (abandon WAIT status, RFU)
28 (abandon PINTE, RFU)
49 (2 MHz clock timing is out of tolerance)
53 (ground)
54 (assert SLAVE CLR*, add pullup)
55-56 (need pullups)
57 (relocate DIG1*, add pullup)
58 (relocate FRDY*, disassert sXTRQ*)
60 (needs pullup)
65 (move FRDY* to this NDEF line)
66 (move DIG1* to this NDEF line)
67 (PHANTOM* fake open collector, add diode and pullup)
96 (sINTA jumpers)
97 (sWO* gates data bus drivers, Rev P update)
98 (abandon SSTACK*, add pullup for ERROR*)
99 (maybe)

The following are unsupported bus pins (features) of the bus in the Sol:

4-11 (vectored interrupts)
12 (non-maskable interrupt)
13 (power-failure)
15-17, 59, 61-64 (extended addressing)
58 (sixteen-bit data transfer request)
98 (bus error).

PREVIEW OF PART 2

Enslave the Sol

In the next part of this series, I will examine the changes necessary to let you disable the Sol's 8080 CPU and allow another CPU board to take over as permanent bus master. The rest of the Sol (video, I/O ports, RAM, ROM) would remain available to the new processor. I think this may be done by flipping a switch so that you can always go back to the 8080 processor to run old software.

If you have some time to look into this, please send me your ideas. Suggestion: look into disabling drivers that control the permanent bus master signals, such as portions of U77. One problem is that the dot clock uses a portion of U77, too. We'll have to swap the gates with some other chips, so the dot clock can remain for on-board use, while the bus signals are disabled. The 8080 will have to be put on HOLD so its internal address and data drivers go into the high-impedance state.

Some other problems need to be considered at the same time: 16-bit I/O port addresses, 24-bit memory addresses, and the 16-bit data bus transfers.

Extended I/O Addressing

The 8080 microprocessor puts I/O port addresses onto both the A0-A7 lines and the A8-A15 lines. The Sol uses this fact by decoding the A8-A15 lines when looking for an I/O port address. But the standard now defines A8-A15 as the high order side of a 16-bit I/O port address. A note in the standard warns against using the I/O address the way Sol does, because it complicates expansion to extended I/O device addressing. How do we correct this? We'll look at the problem in the next part of the series.

Extended Memory Addressing

The Sol only considers the basic 16-bit address bus (A0-A15). The standard now allows 24-bit memory addresses. If Sol is to exist as a slave device in an extended memory system, Sol will have to decode the high order 8 bits of address (A16-A23) when deciding if it should respond to a memory cycle on the bus. In the next part of this series, we'll consider extending the Sol's bus to comply with the standard for address lines A16 thru A23.

16-bit Bidirectional Data Bus

In the next part, we'll also examine the problems with Sol's bidirectional tied data lines DI00 thru DI07. Even though the Sol is permitted to send only 8-bit transfers in a 16-bit system (by not asserting SIXTN*), the fact that the ED and OD data lines are physically wired together will prevent any other device from sending 16-bit data transfers. We'll look into untying the two sides of the data bus.

End of Part 1 -- To be continued in next issue.

WHAT'S NEW?

The Z-80 upgrade piggy-back is in final preparations for production. Prototype PC boards are being checked for accuracy. Expect an announcement about price and availability in the next issue of Proteus New.

Proteus News is behind schedule this year, but we'll still deliver 4 issues each 12 months to maintain our status as a periodical. Send in your letters and articles, so I don't have to make those thin issues again, please.

CP/M 3.x (also known as CP/M Plus) supports bank switched memory, bigger files, up to 512 megabytes of disk, hashed directory lookup for faster BDOS operation, automatic disk logging when diskettes are changed (^C not necessary), etc. See article in Byte, July 1983.

Here they come: the LSI hard disk controllers. Western Digital, which made floppies economical with their WD1771 chip, has produced the WD1010 hard disk controller chip for 5-1/4" Winchesters. The WD1014 is a companion error correcting code (ECC) chip. Soon we should see single board computers with both hard disk and floppy disk controllers on the CPU.

NEW S-100 BACKPLANE FOR SOL

Extra Reliability and New Feature

Bob Marsh has arranged to have some backplane boards made for exact replacement of your Sol's present backplane. The problem with the existing backplanes is that the TI S-100 connectors used for the 5 horizontal boards have weak finger contacts. As the S-100 boards plugged into the bus warp with heat and age, some contacts fail to make good electrical connections with the edge connectors. This can result in intermittent problems. This is especially true in Sol's that have had the slave boards plugged in and removed many times.

Replacement of the S-100 connectors is virtually impossible, since all 100 solder joints must be cleanly de-soldered or heated simulaneously without damaging the board. The only practical solution is to replace the board as well as the connectors.

Bob suggests that we use Sullins Hi-Rel (high-reliability) connectors, which are advertised for about $4.00 to $4.50 each depending upon quantity, in the California Digital ad in Byte. (Toll free order number: 800-421-5041.) The backplane needs the Imsai type (with .250" spacing) for the 5 horizontal board connectors. The top extender connector can be unsoldered more easily than the horizontal ones, so you can probably salvage your old one. On my Sol they are very high quality connectors, making tight contact, so I'm sure I'll want to save my old one. If you need a new top connector, be sure to order the Altair type, with .140" spacing between the rows of pins. The closer spacing between rows allows easier soldering to the bus traces on the top of the board.

As long as we are making new boards, Bob and I discussed adding a new feature to the backplane. In the large empty space to the side of the bus sockets, Bob is adding pads for the pullup resistor packs we need for the open collector lines mentioned in the article on upgrading the Sol in this issue.

Price of the bare backplane board (you add the connectors and other components) will be $35, available from Proteus within 30 to 60 days after receipt of your order and check. This price includes bare board, instructions, parts list, schematic, and shipping via UPS. If you are out of UPS (United Parcel Service) service area, please add sufficient money to cover up to one pound shipping weight via the carrier of your choice. California residents add sales tax, please. (Visa and Mastercard accepted.) Place your order through Proteus.

If you have intermittent errors which go away when your boards are removed and re-inserted or just moved a bit in the S-100 slot, your problem is probably a faulty S-100 connector. If your system makes errors when it is in one temperature condition (hot/cold) but the problem goes away at the other temperature, you may be experiencing thermal warping of the boards, and better backplane sockets may be needed. Or, if you are afraid that you may have this problem and won't be able to find replacement parts, this is your chance. It may be your only chance to buy an exact replacement for the Sol S-100 backplane. After this first batch is made, we may never be able to have an economic quantity of orders again. Don't miss the opportunity.

WRITE PROTECT FOR HELIOS 5/13/83

by Dave Burton

A couple of weeks ago, disaster struck. My normally faithful Sol showed a Mr. Hyde side, and scribbled garbage over a disk directory. What the...?? But no problem -- I'd just boot one of my backup disks (of COURSE I had a backup). But then it happened again -- another disk trashed! Now I had a real problem. I still hadn't lost any data, but I had no more expendible disks, and I didn't dare boot a disk which I couldn't afford to lose. On most machines, I would just punch out the write-protect hole and not worry; no matter what the stupid CPU did, the disk would be safe. But, unfortunately, the Helios has no write- protect, right? Wrong! It's in there; you just need a way to enable it. So now my Helios has two ugly little switches on the front, and my problem is solved (well, one problem, anyhow; I still haven't figured out why my Sol went berzerk, it seems to be fine now).

The Helios write-protect connectors are the two unused connectors along the top of the Persci main PCB, labled P19 and P20. P19 is for the right drive (drive 1), and P20 is for drive 0. Each connector has 3 pins: 1, 2, and 4 (pin 3 is missing). Pin 2 is ground. If pin 4 is shorted to ground (pin 2), the write and erase circuitry are disabled for that drive. Just run a pair of wires to a front panel SPST switch for each drive; that's all there is to it.

Dave Burton
2317 University Dr.
Durham, NC 27707
Tel: (919) 489-5002

A 77K-RAM SOL
by Dave Burton

This modification adds a simple bank-select memory mapping scheme to the Sol which lets a program switch between the normal Sol configuration, with up to 61K of RAM (plus display memory and Solos), and a 64K RAM mode, with locations C000 hex to FFFF mapped onto a different RAM board.

This allows the Sol to use:

1) The normal 48K RAM below C000 hex, accessed in either "mapped" or "unmapped" (normal) mode.

plus 2) Solos, the display RAM, the 1K internal RAM, and up to 12K of external RAM, addressed from D000 to FFFF (or C000 to EFFF if Solos has been relocated), all accessed in "unmapped" mode.

plus 3) Up to 16K of additional RAM, addressed from C000 to FFFF and accessed in "mapped" mode.

For a total of 76K of external RAM plus 1K internal RAM.

The extra memory can be used for I/O buffers or to keep parts of a BIOS or UCSD Pascal P-code interpreter "hidden", providing more useful program space.

The modification adds a 1-bit output port, FC, to select between the two modes. The PHANTOM* line and IEEE extended address bit A16 are used to select between the two RAM boards which are addressed from D000 to FFFF. When a "1" is output to port FC, all memory references above BFFF cause A16 to go low (active), and the PHANTOM* line to go high (inactive). I call this "mapped" mode. But if output port FC contains a zero (the power-up state), references to Sol's internal memory (RAM or ROM) cause both PHANTOM* and A16 to go low, and references to any other address from C000 to FFFF cause only PHANTOM* to go low.

The "unmapped" 12K memory is intended to be a board which supports the new IEEE extended (24 bit) addressing (or IMSAI-style A16-disable). It should be switched to reside at address 01D000 hex and ignore PHANTOM*. It will thus be disabled for "mapped" mode references to the 8080's high 12K of address space, and enabled in unmapped mode. A Processor Tech 16KRA is fine for this if it is altered to take PHANTOM* from A16, though only 12K will be usable (since the Sol's internal RAM and ROM take up 4K, usually from C000 to CFFF).

Similarly, the "mapped" mode high-16K memory (from C000 to FFFF) will be addressed with A16 low and PHANTOM* inactive (high). It should ignore A16, and be deselected by PHANTOM* low (active).

The other 48K (from 0000 to BFFF) can be addressed either way, since both A16 and PHANTOM* are high during accesses to the low 48K. It must still be strapped to deselect when PHANTOM* (or A16, if you prefer) goes low, since the Sol will still do its famous four cycle remap of Solos on power-up. During the power-up "remap", both PHANTOM* and A16 go low, so the low-48K board can use either one.

Details

Step 1:
You'll need two chips, a 74LS74 dual flip-flop, and a 74S51 dual Schottky And-Or-Invertor. They can be mounted on a small perf-board, piggybacked to two other 14-pin Sol chips (with pins 7 and 14 soldered to the chip beneath and the other 12 pins bent out to the sides), or mounted "dead bug" style, somewhere near the rear of the Sol's main PCB. In my Sol, I piggybacked the 'LS74 to U53 and the 'S51 to U54. If you mount them this way, you should first solder a 4" piece of wire (I like wire-wrap wire) to pin 8 of U53, since you will need to make a connection there later.

Step 2:
Disconnect the following IC pins on the Sol main PCB by removing each IC, bending the specified pins out to the side, and replacing the IC in its socket:

IC	pins	signal description	schematic page
U76	7	(4-cycle "power-up remap of Solos" signal)	X-15
U77	10	(PHANTOM* driver's input)	X-15
U58	8,9	(old sINTA-deselect buffer)	X-16
U35	11	(OUT-FC* line, conveniently already decoded)	X-16
U24	1	(old PHANTOM*/"power-up remap Solos" signal)	X-16

You will be making connections to the disconnected pins later. All OTHER connections to Sol IC pins must be made WITHOUT disconnecting them.

Step 3:
A) Connect a wire from U76 pin 7 to U24 pin 1. This restores the connection needed to remap Solos down to 0000 for the first four memory fetches after power-up or reset.
B) Connect a wire from U58 pin 8 to U22 pin 6. This connection will be used to disable Sol's internal memory in "mapped" mode.

Step 4:
Connect together S-100 pins 59, 61, 62, 63, 64, 15, 17, and 70. The first 7 pins are IEEE extended address lines A17-A23, and pin 70 is ground. We need this change to use 24-address-bit memory in the Sol.

Step 5 (the big one):
Wire up the circuit shown in the schematic; you must...

A) Connect pins 4 and 14 of the 'LS74 and pins 3, 13, and 14 of the 'S51 to +5V, connect pin 7 of both chips to ground, and connect a ceramic bypass capacitor (0.02 to 0.1 microfarads) between pins 7 and 14 of the 'S51.
B) Connect a wire from U53 pin 8 to pins 1 and 2 of the 'S51.
C) Connect a wire from U34 pin 6 to pins 9 and 4 of the 'S51.
D) Connect a wire from pin 6 of the 'LS74 to pin 10 of the 'S51.
E) Connect a wire from U52 pin 15 to pin 1 of the 'LS74.
F) Connect a wire from U35 pin 11 to pin 3 of the 'LS74.
G) Connect a wire from U95 pin 14 to pin 2 of the 'LS74.
H) Connect a wire from pin 8 of the 'S51 to U77 pin 10.
I) Connect a wire from pin 6 of the 'S51 to S-100 bus pin 16.
J) Connect a wire from pin 5 of the 'LS74 to pin 5 of the 'S51, and from there to U58 pin 9.

Step 6:
Make a few simple checks to reduce the likelihood that you have made a wiring error. Verify that...

A) pins 8-13 of the new 74LS74 and pins 11 and 12 of the 74S51 are unconnected. All other pins on these two ICs should be connected to something.
B) All 6 of the Sol IC pins that you disconnected in Step 2 have wires connected to them.
C) The following Sol IC pins are still firmly seated in their sockets despite having wires soldered to them: U22 pin 6, U34 pin 6, U52 pin 15, U53 pin 8, and U95 pin 14.
D) Pin 7 of the 74LS74 and pin 7 of the 74S51 are both connected to S-100 pin number 59 (use an ohmmeter).
E) U77 pin 9 is connected to S-100 pin 67 (use your ohmmeter). This verifies that the Sol's "PHTM" jumper is in place.
F) You can reassemble your Sol and fire it up. It should work normally.

You can now put up to 76K of S-100 RAM in your Sol (or 80K, with 4K unusable). Some possible configurations are:

A) 64K extended-address RAM from 010000F to 01FFFF, plus an old 16KRA board from C000 to FFFF (with PHANTOM*-disable strapped).
B) 64K of old-style RAM from 0000 to FFFF (with PHANTOM*-disable strapped), plus 16K of extended-address RAM from 01C000 to 01FFFF.
C) 64K of old-style RAM modified to take its PHANTOM*-disable signal from A16, plus a 16K board (with PHANTOM*-disable strapped) addressed from C000 to FFFF.
D) like B, but with an old 8K board (addressed from D000 to EFFF or from E000 to FFFF and modified to take PHANTOM* from A16) replacing the 16K board. This, of course, would provide "only" 72K of S-100 RAM (64K in "mapped" mode; 56K plus Solos & a 4K gap in "unmapped" mode).

Special Notes

1) Many S-100 RAM boards allow "write-through PHANTOM*"; that is, rather than disabling the board when PHANTOM* is active, they only disable the output buffers. The S. D. Sales ExpandoRAM is one such board. To use such a board in the upper 16K, you'll have to modify it to behave the way Processor Technology boards do: disabling both writes and reads when PHANTOM* is active. On the ExpandoRAM I, this can be done by bending pin 6 of U7 (a 74LS138) out to the side and connecting a wire from it to the PHANTOM* jumper (E2 and E3). (You'll also need to modify the ExpandoRAM I as described by D. A. Howe on p. 13 of Vol 5, #1, Proteus/News.)

2) If you want to replace the 74S51 with a 74LS51, you may, but you'll need to connect pins 11 and 12 to pin 10. I recommend the Schottky part, though, since an LS output is not really adequate for driving a bus line.

3) If you use (or may someday use) interrupts in your Sol, add a pullup to S-100 line A13 by connecting a 1K resistor between pins 10 and 14 of U22. This will prevent spurious responses from Sol's internal memory during 8080 INTA cycles.

4) If you have relocated Solos to F000 by connecting pins 2 and 12 to pins 5 and 9 on U22, the memory mapper will still work fine, and Solos will still be at F000 -- UNLESS you are also using interrupts in your system! In that case, you should have AB/AD and AE/AC strapped, and you'll have to use something other than pins 9 and 10 of U58 to deselect Sol's internal memory when in "mapped" mode (since that section of U58 is in use). I suggest pins 5 and 6 of U58; this will, however, disable bit 1 of the Sol sense switch (which nobody seems to use, anyhow).

Software Notes

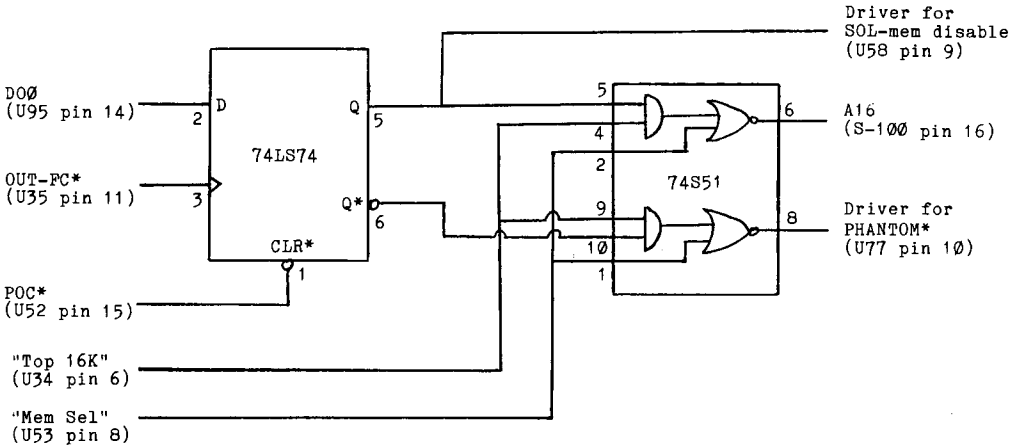
1) The program which changes the mapping flip-flop must reside in the low 48K, of course, and should take care not to zap its own stack if SP points into the high 16K.

2) Although the flip-flop cannot be directly read, it can be easily tested by a program which trys to write to Solos's ROM. E.g.,

```
; returns:
;  A=FF and Z false if mapped onto RAM              (flip-flop=1)
;  A=00 and Z set if unmapped (with Solos at C000)  (flip-flop=0)
;  A=FF and Z set if mapped but no memory at C000   (flip-flop=1)
tstmap: LXI H,0C00H     ;address of Solos
        MOV A,M         ;get old value
        INR M           ;try to change memory
        CMP M           ;did it change?
        RZ              ;no - return Z set (and A=0 if Solos at
                        ; C000, since 1st byte of Solos is a NOP)
        MOV M,A         ;yes - restore the old value
        ORI 0FFH        ;set A=FF (and Z still false)
        RET
```

This would have to be done, for instance, by an interrupt handler which kept a FIFO buffer hidden in high memory, so that it could restore the mapping flip-flop before returning from the interrupt.

See next page for diagram.



Dave Burton
2317 University Dr., Durham, NC 27707
(919) 489-5002

CONSOLE I/O ROUTINES FOR MCVIDEO, INCLUDING SCREEN PRINT

by Charles H. Stembridge

```
*
*    This code contains keyboard input, console output, VIDEO
*    hardcopy and SCREEN scroll speed/stop routines.  This code
*    assumes that the Bob Hogg 24 X 80 display has been
*    installed and the F000 page is in use.
*
*    Hardcopy of the contents of the CRT may be obtained any
*    time the screen is stable by stiking the  PRINT key.
*    This PRINT key must be defined by the user at assembly.
*    An unused key such as the tilde (~) or other may be used
*    or if the number pad adapter kit has been installed a key
*    can be dedicated to this purpose.
*
*    SCROLL speed control may be changed at any time during
*    screen output by striking the numerals 0 through 9
*    (0 fastest and 9 slowest).
*
*    The DISPLAY may be halted at any time by striking the
*    SPACE bar aswell as CONTROL-S.  Display scrolling may
*    be resumed by striking any key.
*
*    The LOAD key causes a JUMP to SOLOS at F004H.
*
*    The MODE SELECT key will cause a CP/M system WARM BOOT.
*
*    The CLEAR key will cause a SCREEN CLEAR.
*
*****************************************************************
```



```
*  CONSOLE INPUT (This is the CPM CONIN routine)

CINP    IN      STATP           ;Get status
        ANI     IRDY            ;Loop for
        JNZ     CINP            ;Character
        IN      DPORT           ;Get character
        CPI     80H             ;MODE SELECT?
        JNZ     NXT
        MVI     A,03            ;Sends CTRL-C to CP/M
NXT     CPI     8BH             ;CLEAR?
        JZ      CLR
        CPI     8CH             ;LOAD?
        JZ      SOLOS           ;Exits to SOLOS at F Page
-->     CPI     0B0H            ;PRINT key? should be changed
                                ;for your designated key
        JZ      HRDCPY          ;PRINT is screen copy key
        RET
CLR     PUSH    B
        MVI     B,0BH           ;CLEARS screen
        CALL    CLEAR
        POP     B
        MVI     A,0DH           ;Regenerates PROMPT
        RET

*****************************************************************
*
*    CONSOLE CONTROL ROUTINE - This routine con-
*    trols the CONSOLE OUTPUT as well as pro-
*    viding single key SCREEN CLEAR, WARM BOOT
*    and SOLOS JUMP capability.  The routines
*    were taken from PROTEUS, Vol. 3, No. 5/6
*    page 25,26.
*
*****************************************************************

CCTRL   CPI     13H             ;CTRL-S?
        JZ      PAUSE           ;The SPACE BAR may also
        CPI     20H             ;be used to temporarily
        JNZ     SCROLL          ;HALT console output;
PAUSE   CALL    CONIN           ;Any key resumes output.
        RET
SCROLL  CPI     3AH             ;This routine regulates
        RNC                     ;dynamically the SPEED
        CPI     30H             ;of SCROLLING.
        RC
        ANI     0FH
        RAL
        RAL
        NOP
        STA     SPEED
        RET

*****************************************************************
*
*    VIDEO hardcopy routine suggested by an article in KILOBAUD
*    MICROCOMPUTING for October, 1980, p. 168., but adapted for
*    the special case of 80 character line length using memory
*    mapped video display.
*
*****************************************************************

HRDCPY  PUSH    B               ;Save registers
        PUSH    H
        CALL    VDADD           ;Get cursor position
        PUSH    H               ;Save address of cursor
        LXI     D,800H          ;800H=0-F800H...Result in HL is
        DAD     D               ;# of bytes past F800 that cur-
        LXI     D,0FFB0H        ;sor is...FFB0=0-50H.  Do mod
SUBB    DAD     D               ;50H math to determine how many
        JC      SUBB            ;characters past line start
        LXI     D,50H           ;cursor is.
```

```
        DAD     D               ;L now has # chars past line
        XCHG                    ;start...now they're in E too
        POP     H               ;Restore addr of cursor
        MOV     A,L             ;Set to beginning of line
        SUB     E
        MOV     L,A
        CC      BUMPIT
        LXI     D,50H           ;Increment by one line
        DAD     D               ;so copy starts at top
        CALL    CHEKIT          ;Have we crossed top of VDMEM
        MOV     D,H             ;Save page start
        MOV     E,L             ;addr in DE
VP2     MVI     B,50H           ;Set up line length
VP3     MOV     C,M             ;
-->     CALL    LIST            ;LIST is address of your CPM
        INX     H               ;LST: device driver
        DCR     B
        MOV     A,B
        CPI     0
        JNZ     VP3
        MVI     C,0AH           ;My CPM LST: device doesn't need
        CALL    LIST            ;a CR; just a line feed.  If yours
-->     MOV     A,H             ;needs both add MVI   C,0DH  and
        CMP     D               ;CALL   LIST at the arrow
        JZ      DUNYET
VP5     MVI     A,0FFH          ;Above VDMEM?
        CMP     H
        JNZ     VP2             ;No
        MVI     A,80H           ;Maybe
        CMP     L
        JNZ     VP2             ;No for sure
        CALL    FIXIT           ;Yes
        JMP     VP2
CHEKIT  MOV     A,H
        CPI     0FFH
        RNZ
        MOV     A,L
        CPI     80H
        RNZ
FIXIT   LXI     H,SCRN
        RET
BUMPIT  INR     H
        RET
DUNYET  MOV     A,L
        CMP     E
        JNZ     VP5
OVER    POP     H
        POP     B
        JMP     CONIN
************************************************************
*
*   PORTS AND STATUS BITS DEFINED
*
************************************************************

SCRN    EQU     0F800H          ;VIDEO SCREEN ADDRESS
STATP   EQU     0FAH            ;CONSOLE STATUS PORT
DPORT   EQU     0FCH            ;CONSOLE DATA PORT
LSTAT   EQU     0F8H            ;LISTER STATUS PORT
LDATA   EQU     0F9H            ;LISTER DATA PORT
IRDY    EQU     1               ;CONSOLE IN MASK BIT
PXDR    EQU     04H             ;PDATA BUSY MASK BIT
ORDY    EQU     80H             ;CONSOLE OUT MASK BIT
LRDY    EQU     80H             ;LISTER MASK BIT
VDADD   EQU     0F137H          ;VIDEO DISPLAY ADDRESS
SPEED   EQU     0FF8BH          ;SPEED CONTROL BYTE
SOLOS   EQU     0F004H          ;USE 0F000H IF YOU WANT
                                ;SCREEN CLEARED ON JUMP
CLEAR   EQU     0F054H          ;SOLOS SCREEN CLEAR ROUTINE
```

CP/M 2.2 BIOS USER AREA ROUTINES FOR SOL/NORTHSTAR

by Bill Loughman

Enclosed is a check in payment for a list of SOL users in my area, as well as an assembler listing of a NorthStar CP/M User Area which may help those trying to run Supercalc, dBaseII, and others.

In the listing, provision is made for several functions not found in the usual CP/M BIOS: GO2XY (position the cursor on a VDM), CAH (clear screen and home cursor), RON and ROF (toggle reverse video on/off), and LFT and RGT (cursor left/right 1 position). Both dBase and Supercalc can do nicely without a Clear-to-End-of-Line, as they provide such functions if you can't.

The bulk of the listing is the delivered substance of the Lifeboat Assoc. CP/M user area, but it has been modified. A smaller but substantial part is the subroutine VTERM, including the functions listed above. By changing 'equates', the whole thing is tailored to fit your CP/M and version of SOLOS. It all fits into NorthStar double density sector 19 (where it has to go).

To use this, you initialize/install Supercalc, e.g., to send an escape-sequence to "the terminal" (which is your own SOL). for example, to place the cursor on row 11, column 11, Supercalc must 'send' this:

```
                ESC   GO2   30H   30H
            (   1BH  ctl-G 0A+20H 0A+20H   )
```

The 20H is required, since row and column numbers less than 32-decimal look like control characters, and CP/M traps some for its own use. Adding a 'bias' or 'offset', then removing it in VTERM, allows us to fool CP/M.

Do remember that row and column number from 00hex, i.e. decimal row 10 is hex row 09.

I value my SOL highly. Other machines are newer, faster, produce nice graphics, and so forth. But I have complete control, and access to about every function in the machine. Sometimes not so easy with many of the "user-friendly" things on the market. Perhaps one day I'll add another machine to wish list -- but you bet that SOL (with all the nice MCxxx upgrades) will be my workhorse for a long time yet.

Sincerely, Bill

WILLIAM D. LOUGHMAN

GENETICIST — 393 GRAVATT DRIVE · BERKELEY, CALIFORNIA 94705

```
0000 * NEWXY.S  :  18 Dec 82 -wd1   11:50PM
0000 * USER (sctr 19 N* DD) for CP/M 2.2 + SOL-20
0000 * ==>> highly self-modifying <<==
0000 * w/ VTERM = 'video terminal' incl GO2XY etc.
0000 * ex XYUSER.S  15 Aug 82  (EXIT1 use: 26 Oct 82)
0000 * ex SOLUSR.S  22 Jul 82  (no LASTO)
0000 * ex N* CP/M USER area, disas 3 Oct 81
0000 *
0000 * VTERM uses SOLOS subr: OCHAR  AOUT
0000 *                    bufs: BOT
0000 *       chngs        bufs: NCHAR  LINE
0000 *
0000 *
0000 USER EQU 0DA00H  56K CP/M 'USER' ORG
0000 ; USER EQU 0BA00H  48k
0000 *
0000 *
0000 * defin ESC seq to VTERM:
0000 *
0000 *   byt1
```

```
0000 ESC EQU 1BH  [ESC] [byt2 .. [ [byt3] [byt4] ]
0000 *
0000 *   byt2
0000 GO2 EQU 'G'  [ESC] [GO2] [byt3] [byt4] = go to X/Y
0000 RON EQU 'N'  [ESC] [RON] = vdm rev on
0000 ROF EQU 'F'  [ESC] [ROF] = vdm rev off
0000 CAH EQU 'C'  [ESC] [CAH] = vdm clear/home
0000 LFT EQU 'L'  [ESC] [LFT] = crsr left 1
0000 RGT EQU 'R'  [ESC] [RGT] = crsr right 1
0000 *
0000 *   byt3,4 = [col +XHIDE], [row +YHIDE]
0000 XHIDE EQU 20H  MUST= 20H or more..
0000 YHIDE EQU XHIDE  ..to fool CP/M for VTERM
0000 *
0000 *
0000 ************************
0000 * SOLOS equ: pick 1 set
0000 *
0000 **********
0000 *  hi MCS  ( WDL version )
0000 MM EQU 0F000H  SOLOS org
0000 MR EQU MM+0F80H  its ram buffers
0000 OCHAR EQU MM+98H  byt in B to vdm at X/Y
0000 VRAM EQU MM+800H  vdm org
0000 XMAX EQU 80  vdm chars wide
0000 YMAX EQU 24  vdm lines high
0000 *
0000 **********
0000 *  lo MCS  ( WDL version )
0000 ; MM EQU 0C000H
0000 ;
0000 **********
0000 *  all PTCO
0000 ; MM EQU 0C000H  lo  ; 0F000H hi
0000 ; MR EQU MM+800H
0000 ; OCHAR EQU MM+98H
0000 ; VRAM EQU MM+0C00H
0000 ; XMAX EQU 64
0000 ; YMAX EQU 16
0000 ;
0000 ************************
0000 *
0000 *  all SOLOS (versions known to me)
0000 AOUT EQU MM+1CH  psudoport out: strips par
0000 AINP EQU MM+22H  psudoport in: keeps par
0000 STAPT EQU 0FAH  gen'l status port
0000 UIPRT EQU MR  user psudoport adrs
0000 UOPRT EQU UIPRT+2
0000 NCHAR EQU MR+08H  cur col= X
0000 LINE EQU NCHAR+1  cur row= Y
0000 BOT EQU LINE+1  txt dsplcmnt
0000 YBIAS EQU YMAX*4  for SOL-20 scrolling
0000 *
0000 *
0000 * gen'l equ
0000 *
0000 PSW EQU 6  $$ rmov if using CP/M ASM or MAC
0000 *
0000 *  verities ->> NO CHNG
0000 XXX EQU 0  buffer in self-mod code
0000 NUL EQU 0  integer zero (XRA A)
0000 FLS EQU 0  boolean false (XRA A)
0000 TRU EQU FLS-1  boolean true
0000 CTL EQU 40H  cntrl ky neg bias
0000 SB EQU 100H  8-bit shftr
0000 RET EQU 0C9H  8080 cod
0000 *
0000 *  SOLOS special keys and codes
0000 BSP EQU 'H'-CTL  non-dstrctv bkspc (input)
0000 DEL EQU 7FH  dstrctv bkspc (inpt)
0000 BAK EQU 5FH  dstrctv bkspc (outpt)
0000 CLR EQU 0BH  clear vdm and reset
0000 HOM EQU 0EH  home crsr
0000 LAROW EQU 01H  crsr left
0000 RAROW EQU 13H  crsr right
0000 MODE EQU 80H  abrt, etc
0000 *
0000 *  SOLOS crsr positioning (outpt)
0000 XCOD EQU 1  [ESC] [XCOD] [col]
0000 YCOD EQU 2  [ESC] [YCOD] [row]
0000 *
0000 *  psudoports
0000 VDM EQU 0  vdm psudoport (XRA A)
0000 KYB EQU 0  kybrd psudoport (XRA A)
0000 *
0000 *  flgs
0000 B1 EQU NUL  byt1 flg
0000 B2 EQU B1+1  byt2 flg
0000 B3 EQU B2+1  byt3 flg
0000 B4 EQU B3+1  byt4 flg
0000 NOKY EQU MODE  n psudoport inpt
0000 *
0000 *  masks
0000 LO2 EQU 3  keep 2 low bits
0000 PAR EQU 7FH  kill parity bit
0000 CRS EQU 80H  crsr on/off
0000 *
0000 *  CP/M
0000 IOBYT EQU 3
0000 *
0000 *
0000  ORG USER
0000 *
0000 * CBIOS jmp vector; data
0000 JMPS EQU $  (keep 11 lines in order)
0000  JMP CINIT  cold init trmnl
0000  JMP WINIT  warm init trmnl
0000  JMP CSTAT  CON: status
0000  JMP CONIN  CON: byt in
0000  JMP CONOU  CON: byt out
0000  JMP LISTR  LST: out drvr
0000  JMP PUNCH  PUN: out drvr
0000  JMP READR  RDR: in drvr
0000  JMP PSTAT  printer status
0000 *
0000  DW LNGTH  len this code
0000 *
0000 * define IOBYT:  list punch reader console
0000 IOCOD DB 94H  10    01   01    00
0000 *
0000  DB 0  rsrvd
0000  DW 0
0000 *
0000 * cold init
0000 CINIT MVI A,NUL  cler status
0000  OUT STAPT
0000  LDA IOCOD  set dflt psudoports for CP/M
0000  STA IOBYT
0000  LXI H,CUSTI  set SOLOS custm outpt
0000  SHLD UIPRT
0000  LXI H,CUSTO
0000  SHLD UOPRT
0000  LXI H,NOKY*SB+NOKY  zap byt inpt bufs
0000  SHLD PRVIN
0000  SHLD PRVIN+2
0000 *
0000 * warm init
0000 WINIT CALL CLEAR  cler vdm
0000  CALL VADDR  locate crsr
0000  RET .  -> CP/M
```

```
0000 *
0000 * ck printer status
0000 PSTAT MVI A,FLS  we're always redy
0000  ORA A  z,nc
0000  RET .  -> CP/M
0000 *
0000 * ck psudoport inpt status
0000 *    in software, 'caus SOLOS also reads the byt
0000 CSTAT LDA IOBYT  CON: status (kybrd)
0000  ANI LO2
0000 *
0000 ISTAT CALL STORE  sv regs
0000 *
0000  MOV C,A  clect correct byt buf
0000  MVI B,NUL
0000  LXI H,PRVIN
0000  DAD B  according to psudoport used
0000  SHLD PNTLI  pnt to it
0000 *
0000  MOV A,M  byt inpt prvsly by AINP?
0000  CPI NOKY
0000  JNZ CSKEY  y
0000 *
0000  MOV A,C  psudoport frm IOBYT
0000  CALL AINP  must read byt to gt status
0000  MOV M,A  so sv any byt
0000  JNZ CSKEY  for nxt tym round
0000 *
0000  MVI M,NOKY  if no inpt (=NUL or =MODE)
0000  MVI A,FLS  tell CP/M
0000  JMP EXIT1
0000 *
0000 CSKEY MVI A,TRU  or say byt 'ready'
0000  JMP EXIT1  (actually already read)
0000 *
0000 *
0000 * input 1 byt
0000 *
0000 CUSTI DB RET  SOL cstm inpt =undfynd
0000  DW 0
0000 *
0000 CONIN LDA IOBYT  CON: fast inpt
0000  JMP INPUT
0000 *
0000 READR LDA IOBYT  RDR: slow inpt
0000  RAR
0000  RAR
0000 ;
0000 INPUT ANI LO2
0000  STA $+4
0000 INPXX CALL ISTAT  await ok
0000  MVI A,XXX
0000  JZ INPXX
0000 *
0000  PUSH H
0000 PNTLI EQU $+1  point to PRVIN byt buf
0000  LXI H,XXX  gt byt svd by ISTAT
0000  MOV A,M
0000  ANI PAR  strip hi bit
0000  MVI M,NOKY  tell ISTAT ok
0000  POP H
0000  RET .  -> CP/M
0000 *
0000 STORE SHLD SAVHL  sv regs
0000  MOV H,B
0000  MOV L,C
0000  SHLD SAVBC
0000  MOV B,C  SOLOS sends frm B
0000  RET
```

```
0000 *
0000 *
0000 * outpt 1 byt
0000 *
0000 CUSTO DB RET  SOL cstm outpt =undfynd
0000  DW 0
0000 *
0000 CONOU LDA IOBYT  CON: psudoport outpt (vdm)
0000  JMP OUTPT
0000 *
0000 LISTR LDA IOBYT  LST: alt1 psudoport outpt
0000  RLC
0000  RLC
0000  JMP OUTPT
0000 *
0000 PUNCH LDA IOBYT  PUN: alt2 psudoport outpt
0000  RAR
0000  RAR
0000  RAR
0000  RAR
0000 ;
0000 *
0000 OUTPT CALL STORE  sv regs, B=char
0000 *
0000  ANI LO2  to vdm?
0000  JZ VTERM  y
0000 *
....  CALL AOUT
0000 *
0000 EXIT MOV A,C  byt 'sent' by CP/M rets in acc
0000 SAVBC EQU $+1
0000 SAVHL EQU $+4
0000 EXIT1 LXI B,XXX  rstor regs
0000  LXI H,XXX
0000  RET .  -> CP/M, flgs=undfined
0000 *
0000 *
0000 * emulate a terminal
0000 *
0000 ESCFL EQU $+1  flg: ESC seq (byt num)
0000 VTERM MVI A,XXX  in esc seq?
0000  ORA A
0000  JZ BYT1  n, but we might want one
0000 *
0000  SUI 2  set flgs to reflect cod
0000  JM BYT2
0000  JZ BYT3
0000 *
0000 * ck byt4
0000 BYT4 EQU $  fall thru, in this mod only
0000 *
0000 * ck byt3
0000 XYFLG EQU $+1  flg: set seq X/Y curs pos
0000 BYT3 MVI A,XXX  byt3, byt4 of crsr X/Y?
0000  ORA A
0000  JNZ GO2XY  y
0000 *
0000  JMP VOUT  n, out byt as-is
0000 *
0000 * ck byt2 (SOLOS uses ctl byts)
0000 BYT2 MOV A,C
0000 *
0000  CPI GO2  start crsr X/Y?
0000  JZ SETXY  y
0000 *
0000  CPI LFT  crsr lft?
0000  JZ GOLFT  y
0000 *
0000  CPI RGT  crsr rt?
0000  JZ GORGT  y
```

```
0000 *
0000  CPI RON  rev vdm?
0000  JZ REVON  y
0000 *
0000  CPI ROF  nrml vdm?
0000  JZ REVOF  y
0000 *
0000  CPI CAH  vdm clear-and-home?
0000  JZ CLEAR  y
0000 *
0000  JMP VOUT  nun of above
0000 *
0000 * ck byt1
0000 BYT1 MOV A,C
0000 *
0000  CPI DEL  del ky? (rubout)
0000  JZ BSPC
0000 *
0000  CPI BSP  bkspc?
0000  JZ GOLFT
0000 *
0000  CPI ESC  start esc seq?
0000  JZ SETESC  y
0000 *
0000  CPI ' '  other ctl byt?
0000  JC VOUT  y, send to SOLOS for intrprt
0000 *
0000 *
0000 * output direct to vdm
0000 *
0000  CALL NOCRS  rmov crsr
0000 *
0000  LDA COLOR  add 'color'
0000  ORA C  to cur byt
0000  XRI CRS  add crsr
0000 *
0000  MOV B,A  put cur byt to vdm
0000  CALL OCHAR  adr dfynd by NCHAR/LINE
0000  JMP VEXIT
0000 *
0000 VOUT MVI A,VDM  forc vdm
0000  CALL AOUT  byt is in B, psudoport in A
0000 *
0000 VEXIT CALL EKILL  kill flgs
0000  CALL VADDR  gt/sv crsr adr
0000  MOV A,M  put crsr there
0000  ORI CRS
0000  MOV M,A
....  JMP EXIT
0000 *
0000 *
0000 * VTERM subr
0000 *
0000 * set-up X/Y crsr seq
0000 SETXY MVI A,XCOD  byt3 will= col
0000  STA XYFLG  (incrs to YCOD on nxt)
0000  MVI A,B3  warn: 3+ byt seq
0000  JMP ESET
0000 *
0000 * set-up ESC seq
0000 SETESC MVI A,B2  y, nxt= byt2
0000 *
0000 ESET STA ESCFL
0000  JMP EXIT
0000 *
0000 * kill ESC seq
0000 EKILL MVI A,FLS  kill..
0000  STA ESCFL  ..ESC seq
0000  STA XYFLG  ..XY seq
0000  RET
0000 *
0000 * keep trak of byt in vdm
0000 VADDR CALL VDADD  crsr adr
0000  SHLD CADDR
0000  RET
0000 *
0000 * remov crsr
0000 NOCRS LHLD CADDR  gt crsr adr
0000  MOV A,M
0000  ANI PAR  kill crsr
0000  MOV M,A
0000  RET
0000 *
0000 * locate crsr in vdm
0000 VDADD PUSH PSW
0000  PUSH B
0000  PUSH D
0000 *
0000  LDA NCHAR  =X
0000  MOV C,A
0000  LDA LINE  =Y
0000  MOV L,A
0000  LDA BOT  =Ybias for scroll
0000  ADD L
0000 *
0000 VD0 CPI YMAX
0000  JC VD1
0000 *
0000  SUI YMAX
0000  JMP VD0
0000 *
0000 VD1 LXI H,VRAM-XMAX
0000  LXI D,XMAX
0000 *
0000 VD3 DAD D
0000  DCR A
0000  JP VD3
0000 *
0000  MOV E,C
0000  DAD D
0000 *
0000  POP D
0000  POP B
0000  POP PSW
0000  RET .  HL= crsr adr in vdm
0000 *
0000 * clear vdm; home crsr
0000 CLEAR MVI B,CLR
0000  XRA A  =vdm
0000  CALL AOUT
0000  MVI B,HOM
0000  JMP VOUT
0000 *
0000 * position the crsr at col=X row=Y
0000 GO2XY PUSH PSW  sv XY cod
0000  CALL NOCRS  kill crsr
0000  POP PSW
0000 *
0000  CPI YCOD  which: X or Y?
0000  JZ GOY  want Y
0000 *
0000 GOX INR A  set for row nxt time
0000  STA XYFLG
0000  MVI A,B4  inform GO2XY
0000  STA ESCFL
0000 *
0000  MOV A,C  set col
0000  SUI XHIDE
0000  STA NCHAR
```

```
0000  JMP EXIT
0000 *
0000 GOY MOV A,C  set row
0000  SUI YHIDE
0000  ADI YBIAS  n accidntl scroll
0000  STA LINE
0000  CALL EKILL  kill flgs
0000  JMP VEXIT  sho crsr
0000 *
0000 * move crsr left or right
0000 GOLFT MVI B,LAROW
0000  JMP VOUT
0000 GORGT MVI B,RAROW
0000  JMP VOUT
0000 BSPC MVI B,BAK  dstrctv bkspc
0000  JMP VOUT
0000 *
0000 * clect nrml/invrs vdm
0000 REVON MVI A,NUL  rev vdm
000$  JMP $+2
0000 REVOF MVI A,CRS  nrml vdm
0000  STA COLOR
0000  CALL EKILL  kil flgs
0000  JMP EXIT
0000 *
0000 *
0000 * bufs
0000 CADDR DW 0  vdm adr of crsr
0000 COLOR DB CRS  z= rev vdm; CRS= normal vdm
0000 *
0000 * prv byt inpt frm psudoports:
0000 PRVIN DB 0  0= kybrd
0000  DB 0  1= serial
0000  DB 0  2= parallel
0000  DB 0  3= custom
0000 *
0000 LNGTH EQU $-JMPS  len this CBIOS
0000 *
0000  END .  ZZzz
```

U C S D P A S C A L F O R S O L / H E L I O S

by Dave Burton

I have just reached tentative agreement with SofTech Microsystems to distribute an enhanced version of UCSD Pascal version I.5 for the SOL. I.5 is an old (pre-SofTech) version, lacking some of the features of the latest IV.1 version (e.g., Fortran) but I have fixed a few bugs and made some enhancements (including full error traceback dumps for easier debugging of Pascal programs).

The package will include the basic P-system and OS, the Pascal and (probably) Basic compilers, macro assemblers for the 8080 and Z80 (the P-code interpreter is written in Z80 mnemonics), a screen editor, YALOE (a simple character-oriented editor much like the RT-11 editor or a subset of RDOS's NSPEED editor or TECO), filer, linker, disk formatter, and assorted minor utilities.

The system will come on Helios format disks. Two versions will be included with every purchase. The "unmapped" version boots and runs on any unmodified SOL with RAM from 0000-BFFF and from D000 to FFFF. It is a fully working system, but it suffers from the usual problem of UCSD Pascal systems -- memory shortage. There is not enfough memory to compile and link really large programs (like the compiler itself), and you must usually use the compiler in "swapping" mode (which slows it down considerably).

The "mapped" version runs only on a SOL which has been modified according to my instructions in "A 77K RAM Sol" (also in this issue of Proteus News). It "hides" the BIOS, SOLOS, and the screen memory from the address space of UCSD Pascal programs, leaving much more usable memory; more, in fact, than you are apt to see on any other 8-bit computer running UCSD Pascal.

By the time you read this, I should have recieved final approval from SofTech. The price will be around $200 including manual, bootable disks, source code(!) for the P-code interpreter & BIOS, and SofTech's royalty (note that the package will NOT be supported by SofTech, however). Source code for most of the rest of the package will also be available, for an additional $35 per diskette (probably 4 diskettes to get everything).

David A. Burton
2317 University Drive
Durham, NC 27707
(919) 489-5002

...ON HELP NEEDED TO INTERFACE REAL TIME CLOCK

Dear Sir,

I notice from reading back issues of the SOLUS News recently that you were the editor of this newsletter, and wondered if you could be of any assistance with a few inquiries I have regarding the SOL computer.

First of all, does the SOL User's Group still exist over there, and if so, is it possible for me to join and /or purchase tapes from the software library? I am particularly interested in the tape version of ALS8, as it is impossible to purchase one over here despite many attempts.

Secondly, I have been trying to interface National Semiconductor's 58174A Real Time Clock to the SOL and have been having trouble with it. I believe timing is the main problem and wait states need to be generated, but so far all attempts to do this have met with failure. Has anybody that you know of managed to design an interface for this chip to the SOL, and would they be willing to let me have the circuit involved, as I am most anxious to get the clock working. I have written the software to display the time and date on listings and did have an MM58174 that worked for awhile until the ambient temperature rose, so I was able to get the software working, and would be willing to swap it for the appropriate hardware. I have the clock wired as part of memory at the moment, and would prefer to use it this way rather than have to use input/output ports.

I hope you can be of assistance in these matters as there is no SOL User's Group over here, and most people haven't even heard of the SOL.

Thanking you in anticipation,

Yours faithfully,

Bruce D. Bull

Box 127,
Woodville.5011
Sth.Aust.
26th March 1983

...ON COMPUPRO RAM 17 MODIFIED FOR SOL AND REQUEST FOR HELP WITH TARBELL/HELIOS DISK

Dear Stan:

Keep up the excellent work, blah blah blah . . . it takes everyone to make the effort to write sometime to keep information going to those people that just don't know the difference between their AND and OR gates but here goes for me.

Just home from the computer fair; the deal of the day for anybody's overheated computer was the CompuPro RAM 17. The only thing to do before plugging it in is remove U5 (the 25LS2521); use it in your next project -- very handy! Next, set the switches to enable: All 64K of RAM, NORMAL MODE, GLOBAL MODE. Then remove U3 (one of the 74LS138's), the chip enable decoder for the uppermost 16K of memory. Bend up (or cut) pins 14 and 15 so that when the I.C. is reinserted, they will not go into the socket. By doing this, the two chip enable lines (CE C8, CE C0) won't ever go ACTIVE LOW (0v) because of the pullup resistor SR2, so your Sol's memory can think for itself without any interference. This also gives you two 2K x 8 CMOS static RAM for spare or test use.

I would like to hear from someone (in Calif. or the Bay Area?) that has made the conversion to his Sol (listed in Volume 1, No. 5, by Ron Parsons), regarding the Tarbell single-density disk-controller board interface for running C/PM. I would like to know if both can be run at once, i.e., reading a C/PM file to memory under C/PM control, then switching to PTDOS control and writing it from memory to a PTDOS disk, using only a HELIOS II? Any information regarding a BIOS that software supports the fast-seek mode of the PerSci drive would save the day.

In closing, thank you for the opportunity to share.

Sincerely,

Jim Tittle

4/15/83
178 Thomas Way
Pittsburg, Ca.
94565
(415) 798-8790

[Ed. reply to Jim Tittle: I think Jack Kinney at UCLA Computer Science Dept, 3413 Boelter Hall, Los Angeles, CA 90024, may be able to give you some advice on the Tarbell multi-plexed controller modification.]

...ON FINDING THE 6574 CHARACTER GENERATOR

Dear Stan,

First of all, put my name on your list for the backplane board. I plan to keep my SOL until it won't go no more. You're right that this might be the only chance to stock this vital part.

Secondly, if Gardner Bride is looking for the 6574 character generator ROM, he should order the MCM 66740P. I got one a month ago from Jade Electronics, and it is an exact replacement.

Ron Burford

Ron Burford W88KFD
7074 Constantine Ct.
Mentor, Ohio 44060

...ON A FAST IMPLEMENTATION OF CP/M 2.2 FOR HELIOS

Dear Helios Owners,

There is a CP/M version 2.2 available for the HELIOS disk system. There are several companies in the Washington D. C. area who use it daily, and have been using it for many months. I being a person who grew up in the world of Processor Technology could not take the unbelievably slow speed of Lifeboat's CP/M for Helios. Therefore I began the job of creating one. It works, and has been working for many months. I feel that it is as fast as it can be, and as reliable as I can make it.

How fast is it? Lets use the disk to disk copy program as an example. This program will format the output diskette in under 70 seconds, perform the disk to disk copy in 35 seconds, and verify that the copy worked by performing a disk to disk compare in 45 seconds. The total time for the complete format, copy and verify takes less than 2 and 1/2 minutes. I am not sure of the numbers but I think lifeboat's format takes about 9 minutes by it self.

Yes, I do have a version of CP/M 2.2 for Helios but it is not in a state that I am ready to market. There are no user's notes written, and a few enhancements that have been on the back burner for quite some time that should be added. Items like the ability to read lifeboat format floppies using this CP/M, and a user oriented interactive configuration program to setup printers, and certain flags like number or retries on a disk operation, and whether read after write should be turned on.

So why did I send this letter? I would be willing to make all of the necessary changes and whip this software into a sellable state if there is enough interest in it. I estimate that the cost per copy if less than 50 people want it will be approximately $250, because I will have to pay list price for the CP/M, and manuals from Digital Research. This cost will drop to about $175 if there are more than 50 people interested.

There is a large amount of software that works very well with under CP/M. Microsoft's Basic, and Fortran compilers work. With a bit of setup WordStar works very nicely on a Sol, I prefer to use the Sol for editing because of the speed with which WordStar updates the screen.

If you are interested please send me a letter of intent as soon as possible (DO NOT SEND MONEY) and I will figure up an exact price based on the number of responses. If there is very little interest I might just offer it as-is. I will inform all interested parties, either through the newsletter or by mail, of my intentions and the exact price as soon as I can.

Sincerely,

Lewis Bean

Computer Problem Resolutions
19524 Crystal Rock Drive. #23
Germantown, Maryland 20874

...ON SCREEN PRINT ROUTINE, BIOS FOR MCVIDEO AND MCSOLOS, CCS DISK CONTROLLER

Dear Stan,

Enclosed is my check for renewal of my subscription to PROTEUS. Add my thanks to those of other subscribers for the good work you are doing.

Also enclosed is the source code for a screen copy routine to be added to a CP/M bios (or to PTDOS I suppose) which will allow hardcopy to be made on a printer of the contents of the display screen any time it is not scrolling. It will need to be entered into your bios and reassembled using Digital Research's ASM.COM or any other assembler. I have used it with CP/M 1.4 and 2.2 and with the VISTA disk controller for 5.25 inch disks and the California Computer Systems disk controller for 8 inch disks. I hope it will be of use to someone.

I have adapted CP/M 2.2 to run on the SOL with McVideo and McSolos using the CCS 2422 disk controller driving 8 inch disks in double density. While it was not difficult it was time consuming. If anyone would like to save themselves this effort for their SOL I would be glad to help. Just write to me at the above address.

One additional note of interest. The CCS controller will not read double density tracks at the standard SOL speed of 2Mhz. Remember, the SOL has jumpers which allow changing the clock speed to as high as 2.8Mhz. I purchased an 8080A-1 and tried the higher speed. The SOL ran fine but there were some random screen content changes while editing files (not in the file content - just in the display). However, I found that the original (2Mhz) 8080A which came with my SOL ran at 2.8Mhz just fine! As a result my SOL is now operating at the higher speed and reads the double density disks with no problems.

Sincerely,

Charles H. Stembridge

2427 Frances St.
La Crescenta, CA
91214

...ON HELP NEEDED TO MOVE PTC BOARDS TO HIGHER ADDRESS

Dear Stan:

A couple of items.

First, enclosed is my check for $30.00 to cover my 1983 membership.

Second, keep up the good work. You are doing a fine job. I am now a member of several organizations, and PROTEUS still ranks with the best.

Third, I'm considering making some adjustments to my machine. I would like to move all the PTC boards to a higher address. I still use some software and would like to keep the VDM and the tapes. I seem to remember something about a modified ROM for SOLOS/CUTS. Can you give me some brief observations?

Sincerely

John E. Breden
921 Waterview Cir.
Richardson, TX 75080

[Ed. reply to John Breden: It sounds like you have an S-100 system with PTC boards, rather than a Sol. The relocated ROM was made for the Sol, specifically. If you want a ROM for your VDM/CUTS system, I suggest you contact Bob Hogg at Micro Complex to see if he can help. His address is in my reply to Merle Bowen's letter in this issue.]

...A COMPLAINT ABOUT THE DUAL PERSONALITY MODULE

I recently purchased and installed the video upgrade and corresponding Dual Personality Module for my Sol-20.

After some initial problems getting the pins to make proper contact, the video display has worked reliably and well. I am currently rewriting many of my programs to take advantage of the added display space.

I am not nearly as pleased with the new Dual Personality Module -- it does not execute custom user output routines. This has forced me to consume much time rewriting interface routines. What I really want to know is can get my present Prom replaced with one that has the old pseudo-output port 3 custom user routine?

Sincerely,

Merle L. Bowen
5825 Keith AVenue
Oakland, CA 94618

[Ed. reply to Merle Bowen: Bob added some new features in the latest version, such as emulation of a Televideo 950 terminal, so some more Solos features had to be deleted. Bob assumed that most users would be selecting I/O re-direction through the disk operating system rather than through the Solos ports. He will reprogram your module to the old version if you wish. Please contact Bob Hogg at Micro Complex, 25651 Minos St., Mission Viejo, CA 92691, (714) 770-2168.

...ON McVIDEO JITTER, EPSON MX80 DOWNTIME.

Dear Stan,

Enclosed my check for $30. for 1983 subscription. PROTEUS remains one of my favorite publications.

It is now nearly a year since I had my 24 x 80 upgrade made by McHogg. Generally I have been happy with it, but there is a definite "jitter" to the screen that was not there before. Two letters to McHogg re this have gone unanswered....which has been a disappointment. I have decided to live with the jitter as it seems pointless to write again. He just doesn't seem to answer!

I have an Epson MX80 printer, which ceased to function about 10 days after the warranty expired. (Incidentally, the article in PROTEUS helped immensely in interfacing it). The Epson repair depot fixed it, sent it back to me, and it went down again. I returned it for another repair, and Epson claims it's the SOL that's destroying it! I don't see how that's possible, as it's only TTL driven. However, there's the problem. Anybody else got the same one?

It's always very interesting to get PROTEUS. Keep up the good work.

yours truly,

Jim Jackson
POB 730
Princeton, B.C..

[Editor's reply: I haven't noticed a jitter, although a flicker is perceptable with the 13 line scan. When I switch to the 12 line option, the flicker isn't noticable on my monitor. You should try a friend's video monitor to see if it is in the computer or the monitor. Borrow a good one with high bandwidth and long-persistence phosphor.

Bob is very busy maintaining computer systems, designing the Z-80 upgrade for Sol, manufacturing his Sol upgrades and the Phase Lock II disk controller, etc.. He just doesn't have time to answer every letter. Bounce your comments and questions over to Proteus, and I'll do my best to get answers we can all use.]

BARGAIN PRICE ON 1979 ISSUES

WE'RE OVERSTOCKED ON 1979 VOLUME OF PROTEUS NEWS

Any one wanting PROTEUS 1979 can have all the issues for $1.50 (in continental U.S.) to cover the cost of sending it UPS.

UNCLASSIFIED ADS

For sale: N. S. Controller DD, 2 SS DD Drives, 16 K RAM, Sol Micro. $595, Argonaut Computer Center, 1104 Buchanan Road, Antioch, CA 94509.

WANTED

Cassette Electric Pencil II program to run matrix printer from SOL-20 48K.
7x9 Matrix Printer to be run by above.
Contact with someone who has converted SOL/Electric Pencil data tapes to IBM PC.
Walter Jessel, 1500 Bluebell, Boulder, CO 80302
(303) 442-5757

FOR SALE

North Star Controller DD, 2 SS DD Drivers, 16k RAM, Sol Micro
All for $595.00
Argonaut Computer Center
1104 Buchanan Road, Antioch, CA 94509 (415) 778-2595

For Sale: Sol-20, Solos Module, 48K Dynabyte Memory, Dual North Star Single Density Floppies, NS Dos 2.3, NS Basic3.6, SolStar word processor, Games, CP/M Distribution Diskette, Spare floppy drive, 14" Javelin monitor, 50 diskettes, complete manuals for all of the above. System runs like a top when warm, occassional startup problem when cold. $600.

Richard B. Kern
Richard B. Kern
314 W. Howe St.
Seattle, WA 98119
(206)284-1558

FOR SALE

VDM-1 S-100 board - $50
PARA-SOL hardware debugger - $30
complete back issues SOLUS News/PROTEUS - $20
original manuals and cassettes for

BASIC5	Extended BASIC (w/ CP/M patch)	PILOT
GamePac 1	GamePac 2	TREK80
EDIT	8080 CHESS	DEBUG
FOCAL	ALS-8	SOLOS/CUTER
PolyMorphic BASIC patched for SOLOS		Music System

$5 each, everything for $25

Contact: Bob Stek
19 Mayfield Road
Regina, Saskatchewan
S4V 0B7 Canada
(306) 352-7184

SOL20 WITH 48K, HITACHI MONITOR, DISCUS I, 8IN DISK. CP/M, DISKATE, ECBASIC, BASIC V, VIRTUAL BASIC, MBASIC, BASIC-E. WDSTR, CHESS, TRK-80, ETC. SOME CP/M U.G. SFTWRE. ALL MANUALS, EXCEPT WDSTR, PROTEUS ISSUE 1 THRU CURRENT.. $980 FIRM F.O.B. OXNARD. DICK LINDER, 3051 KERN ST OXNARD CALIF 93033. (805)483-6993.

FOR SALE:
SOL 20,Rev.E wt 32K Static Ram,North Star DOS wt Dual Single Density Discs.Including all Documentation.Equipment little used and in excellent condition. It is now surplus to me and I will entertain any resonable offer. Also have an unused Sunny International Co. 8V-18A,16V-2.5A(+&-) assembled power supply (never used) for $49.50.******** Contact J.D.Knight at 3405 Doral Drive ,Waterloo, Iowa 50701 or Phone 319-233-6123.

FOR SALE

2 ea. Sol-20 (Rev. E.) with Solus module ≠ 32 KRA ≠ 16 KRA boards,..Mint Condition.. All manuals, original package, E C Basic. One set users group publications, Vol 0 - No. 0 to current. .. All offers considered.
M. E. Price, P. O. Box 790, Newark, Ohio 43055
Phone (614) 345-6025... 12 PM - 5 PM EDT.

FOR SALE

SOL-20, Rev. D upgraded by PTCo. to Rev. E, with 48K RAM, all manuals, $750. Memory boards are PTCo. 16KRA and 32KRA. Spare set of memory chips (4108's) for the 32K board. Software includes ECBasic, GamePacs 1 and 2, and FOCAL. Contact:

Charles H. Stembridge
2427 Frances St.
La Crescenta, CA, 91214
213-248-5103 evenings

T A B L E O F C O N T E N T S

EDITORIAL: WHERE DO WE GO FROM HERE? by Stan Sokolow.....1
FUTURE SHOCK or ARE 16 ADDRESS BITS ENOUGH by Stan Sokolow...1
SMALLTALK-80: THE LANGUAGE FOR THE NEXT ERA? by Stan Sokolow.3
THE IEEE 696 STANDARD AND SOL - PART 1: COMPLIANCE AND
UPGRADING AS A BUS MASTER by Stan Sokolow.....4
WHAT'S NEW - EDITOR.....12
NEW S-100 BACKPLANE FOR SOL - Editor.....12
WRITE PROTECT FOR HELIOS by Dave Burton.....13
A 77K-RAM SOL by Dave Burton.....13
CONSOLE I/O ROUTINES FOR MCVIDEO, INCLUDING SCREEN PRINT by
Charles H. Stembridge.....15
CP/M 2.2 BIOS USER AREA ROUTINES FOR SOL/NORTHSTAR by
Bill Loughman.....16
UCSD PASCAL FOR SOL/HELIOS by Dave Burton.....20
LETTERS TO THE EDITOR:
...ON HELP NEEDED TO INTERFACE REAL TIME CLOCK by B.D.Bull.20
...ON COMPUPRO RAM 17 MODIFIED FOR SOL & REQUEST FOR HELP
WITH TARBELL/HELIOS DISK by Jim Tittle.....21
...ON FINDING THE 6574 CHARACTER GENERATOR by Ron Burford..21
...ON A FAST IMPLEMENTATION OF CP/M 2.2 FOR THE HELIOS by
Lewis Bean.....21
...ON SCREEN PRINT ROUTINES, BIOS FOR McVIDEO AND McSOLOS,
CCS DISK CONTROLLER by Charles Stembridge.....22
...ON HELP NEEDED TO MOVE PTC BOARDS TO HIGHER ADDRESS by
John E. Breden.....22
...ON A COMPLAINT ABOUT THE DUAL PERSONALITY MODULE by
Merle Bowen.....22
...ON MCVIDEO JITTER, IPSON MX80 DOWNTIME by J. Jackson....22
BARGAIN PRICE ON 1979 ISSUES OF PROTEUS NEWS.....23
UNCLASSIFIED ADS.....23

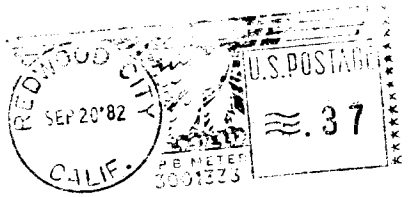
P R O T E U S / N E W S

A news journal for owners and users of Processor Technology Corporation computer equipment. Published by Proteus, 1690 Woodside Road, Suite 219, Redwood City, California 94061-3483,USA, telephone (415) 368-2300.

Submit items for publication to Proteus News, Attn: Stan Sokolow, 1690 Woodside Road, Suite 219, Redwood City, California 94061-3483, USA. Please make submissions as camera-ready as possible by using a fresh, black ribbon and typing single-spaced.

FROM:
PROTEUS
1690 WOODSIDE ROAD, SUITE 219
REDWOOD CITY, CALIFORNIA 94061-3483
USA

FIRST CLASS MAIL



FIRST CLASS MAIL

James D. McElroy
2826 Crest Ave. North
Allentown, PA

18104

PROTEUS / NEWS

AN INDEPENDENT NEWSLETTER FOR OWNERS AND USERS OF PROCESSOR TECHNOLOGY CORPORATION COMPUTERS

FORMERLY SOLUS NEWS

1983 PUBLISHED BIMONTHLY BY PROTEUS, 1690 WOODSIDE ROAD, SUITE 219, REDWOOD CITY, CA 94061, USA

VOL. 6 #2
SINGLE ISSUE...$7.50 (US)
SINGLE ISSUE...$9.50 (FOREIGN)

SOL AND THE IEEE 696 STANDARD

Part 2: Upgrading the Sol, Compliance of Memory and Peripherals

In the first part, we examined the extent to which the Sol complies with the IEEE 696/S-100 bus standard. We saw that the standard was defined in such a way that most of the Sol's design complies and only a few signals had to be moved because of conflicts. Some of the conflicts only happen if you try to use fancy things, such as multiple DMA devices (temporary bus masters) all trying to take the bus concurrently.

In this part, we'll look at the question of extending the Sol to handle the features of the bus that it doesn't yet implement: extended memory addressing, extended I/O addressing, TMA arbitration, and so on. We'll also look at the Helios disk system and Processor Technology memory boards.

That first part really wore me out and time is short for this part, so I may leave out some details. I'm counting on you to let me know and fill in the gaps.

EXTENDED MEMORY ADDRESSING

The IEEE standard now provides for 24 bits of address lines, whereas the Sol was designed with 16 bits because that is all that the 8080 microprocessor can address. With 16 bits, you can address 64K of memory. With 24 bits, you can address 256 times that much, amounting to about 16 megabytes. Can the Sol be modified to address more than 64K?

Sure. It is possible to add circuitry to let the programmer output an 8-bit byte to a certain I/O port, where it would be latched (stored) onto the extended 8 address lines. The Sol's address decoder could likewise be modified to compare the extended lines when deciding if the bus is addressing memory space in the Sol's 4K block (normally at C000, optionally at F000 with the DPM module we sell). This would let the Sol's memory respond only to addresses in one of the 256 possible 64K segments. The particular segment would best be the first one or the last one if only one were hardwired into the circuit, but it could also be designed to be switch selectable with 8 DIP switches or even latched by program control.

But then the question comes, what would you do with more memory? Dave Burton in a previous Proteus News issue has given us his method for extending the Sol to have an extra bank of 16 K for hiding the p-code interpreter in his implementation of UCSD p-system version 1.4.

(continued on page 7, right column)

MEET ADA, THE NEW PROGRAMMING LANGUAGE

If you've been reading the computing journals over the past few years, you may have come across the ongoing story of the U.S. government's project to develop a standard language for programming large defense systems. Beginning in 1974, the Department of Defense (known as DoD to the in-group) has been working on the requirements, specifications, and implementation of a computer language for the large, real-time computer systems that control missiles, radar networks, and smart weapons of all types. This effort has culminated in the adoption of the Ada language standard on February 17, 1983, by the American National Standards Institute (ANSI).

There has been controversy surrounding Ada throughout the process. Some people claim that the language is too big, having too many features that are difficult to implement and validate. They worry that an obscure compiler bug will result in a program error that causes a weapon system to go wild and start World War III or cause some other loss of life and property. These critics prefer the clean simplicity of Pascal or Modula.

Other experts have been disappointed by the project's rejection of their favorite language feature, in essence saying that the language isn't big enough.

Nevertheless, in spite of the criticism, Ada is here to stay. The DoD has decreed from the highest levels that Ada shall be used on all mission-critical applications after January 1, 1984. There are now five defined mission-critical areas: intelligence systems, cryptologic systems related to national security, command and control of military forces, computer systems that are an integral part of weapon systems, and other applications critical to the direct fulfillment of military or intelligence missions.

This doesn't mean that all of the zillions of dollars worth of COBOL programs will be scrapped. COBOL will probably still be used for the business of running this arm of government, such as doing the payroll, although that may even be interpreted as mission-critical one of these days. FORTRAN may still be the language of choice on scientific research projects under government contract since the program libraries in FORTRAN are immense.

But for the new high-tech weapons, communication networks, and such, the control programs will probably be written in Ada for the foreseeable future. (There will be exceptions made by the DoD for contractors who must use computers for which there is not yet an Ada implementation, but this will be temporary.) The goverment will specify Ada, and the contractor will have to comply. And this means that the entire high-tech defense industry will be gearing up for Ada programming.

(continued on page 4, left column)

CLARIFICATION OF DUAL PERSONALITY MODULE

From time to time we've received questions about the various configurations of the McDPM Dual Personality Module. The McDPM allows you to modify your Sol to permit up to 60K of contiguous RAM memory. The standard Sol has Solos ROM and the Sol's screen and scratchpad RAM addressed in the 4K block beginning with C000. This only allows 48K of contiguous memory below it. The McDPM allows you to switch the base address from C000 to F000, thus allowing the 60K space below F000 to be available to your system programs.

In addition, the McDPM comes with a new version of Solos (which I've dubbed "McSolos") that eliminates the tape commands and some other seldom used features and instead provides support for the 24x80 screen module. It also can include a bootload command for booting your floppy disk.

The usual configuration to buy is Configuration A, which comes with two EPROMs programmed with McSolos, one for the C000 mode to be compatible with screen software assuming a standard Sol, and one for the F000 origin mode to allow 60K. The McDPM switches between these two EPROMs at the same time the address is changed from C000 to F000.

Here is a reprint of the catalog description. The full catalog was printed in Proteus News, Volume 5, number 1.

Item M2: McDPM -- Dual Personality Module. $95.00

Replaces Solos personality module and has two EPROMs (2716). Switch on back of module (accessable without opening Sol covers) selects which EPROM is active. When used with the McVideo upgrade (Proteus item M1), the switch also selects Sol RAM, ROM, and video to be at C000 or F000 address, and sets video display form (16x64 versus 24x80). Programmed with various versions of Solos (see below).

The new McSolos is an altered version of Solos which replaces the tape I/O commands with other commands, such as Test Memory, Dump Ascii, Move Memory Block, etc. It also provides for a selectable auto-bootload on power-on/reset, or bootload on LOAD key. See description of McSolos in Proteus News, Volume 5 Number 1. (Custom versions available, request price quotation.)

Standard configurations:

Configuration A = 2716 EPROMs in both C000 and F000 socket, programmed with same version of McSolos except the C000 origin McSolos provides 16x64 video routines, while F000 McSolos provides 24x80 routines for use with the McVideo mod.

Select this configuration if you want to have identical Solos monitor features in the C and F modes. To use tape routines, you should load them from disk or replace your old personality module temporarily. (Note: a Processor Tech personality module will still work after installation of the Dual Personality module, but a minor change must be made to it first; explained in installation guide.)

Configuration B = Empty socket C000 and board jumpered to accept 9216 masked ROM in C000 socket; you move your original Solos ROM into this. Socket F000 has a programmed 2716 EPROM containing McSolos with video output routines for 24x80 screen.

Select this configuration if you want your Sol to function completely normally with the Dual Personality Module switched to C000 setting, including tape routines, but to talk to the 24x80 screen properly when Sol is relocated to F000.

Configuration C = unprogrammed 2716 EPROMs in both sockets.

Select this configuration if you don't plan to use the 24x80 video upgrade, can program your own 2716's, and want to relocate Sol's address space to F000 for more contiguous RAM space or if you want to do something special in Solos.

When ordering, specify which configuration and which disk controller bootload routine to include in the McSolos:

1. Helios controller.
2. NorthStar. Specify ROM origin and DOS origin.
3. Tarbell single density controller.
4. Versatile Disk Controller (Proteus item M5).
5. McFloppy (NorthStar compatible).

Other controllers boot routines available on special order.

INTERESTING ADVERTISING WE'VE RECEIVED

Here are things you may want to inquire about. If not, at least you didn't have to sort through the junk mail.

Glen Buie, a Proteus member, has a repair shop for Sol and Helios equipment. Doing business as "ProcTech of Texas", Glen can do all repairs, including PerSci drive repairs. Maximum charge for a drive is $200 and for a Sol console $225. He has a flat-rate table for the more common problems, so you won't be hit with a surprise, and for unusual debugging problems his hourly rate is $40/hour. He says his turnaround time is about 6 working days, but recommends that you call him before shipping your machine so he can tell you when to ship for quickest turnaround. He is offering a Special 25% off Labor charges for a limited time. Contact Glen at ProcTech of Texas, 3625 Sunset Lane, Arlington, TX 76016, (817) 275-1015.

Cleo is Here! the flyer proclaims. CLEO is "Computer Listings of Employment Opportunities". It is free to the user, except for your phone call. Use your terminal and modem to access the CLEO listings of job openings. Completely confidential--your job inquiries and applications are known only to the prospective employers you contact, the ad states. CLEO requires a 300 baud, full-duplex, ASCII terminal. Access phone numbers are available in the following area codes: 415, 408, 213, 714, 619. For access assistance: (213) 618-1525.

FileDriver is an integrated collection of file management utilities for CP/M and similar operating systems. Either in menu mode or command mode, you can achieve file operations with a minimum of keystrokes. Utilities include: Archive, Attr, Cdump, Copy, Csub, Custom, Default, Erase, ListF (lists file directory info in various formats), Print, Rename, Verify, Wait, ... Features include: command line interface allowing multiple commands per line, can be run from submit file, input/output redirection, takes commands from text files, help facility to teach you command line construction, create your own commands, clear error messages without need for a manual, requires no modification to CP/M and makes no modifications to CP/M, familiar syntax, reasonable price ($69). For more information, contact Dunbar-Ridge Corp., 102 Sterling Court, Syosset, NY 11791, phone (519) 496-4431.

Micro-Grip is an inexpensive friction feed for Epson MX-70, MX-80, and RX-80 printers which only come with forms tractors. This device reportedly installs easily, is rugged, and still allows you to use the forms tractor. Now you can feed your cut sheets of letterhead, use roll paper, etc., without upgrading to the more expensive versions of Epson printers that provide friction and tractor feeding. Micro-Grip III costs $39.95 plus $2.00 shipping, and is available directly from Micro-Grip, Ltd., 3164 Dumbarton Avenue, San Bernardino, CA 92404, phone (714) 864-6643.

Computerized Greeting Cards (continuous forms with original pen-and-ink art for Christmas, wedding, anniversary, invitation, etc.) are available from CompuCards, P.O. Box 894, Stone Mountain, GA 30086, phone (404) 299-0173. Christmas card prices are $9.95 for 20 cards and envelopes, $45.00 for a box of 100, or $130 for a box of 300. Cards and envelopes come continuous with tab feed (removable perforated margins for forms tractors). Cards fold to 5.5" by 7.25". For members of Proteus, they are offering a 30% discount on orders over $50 if you mention that you are a Proteus member and read about the discount letter from Lois Schwarzhoff of CompuCards.

ACCESS is a new journal for engineers and scientists interested in problem solving with their microcomputer. Topics include mathematical modeling, statistical analysis, project planning and management, process design, numerical analysis, computer simulation, data reduction, economic analysis, book reviews, software reviews, etc. Subscription rate is $16 for 6 issues. Published by LEDS Publishing Co, Inc., P.O. Box 12847, Research Triangle Park, NC 27709, phone (919) 493-4863.

COMPUTING!, the publishers of "POWER!", a CP/M utility package, have announced version 3.3 with many updates and new features, including password protection. The password facility actually makes the protected files disappear from the directory. Anyone wanting to change or even print them must know the password to put them back into use. They can't spy on the files without the password. The password is not visible to prying eyes either, not even with DDT. The new POWER! costs $169, but an update for older versions is available for $35. Contact COMPUTING!, 2519 Greenwich, San Francisco, CA 94123, phone (415) 567-1634.

COMPUTING! also has announced a PC-DOS version of POWER!, known as MENU/POWER!. It is available for PC-DOS, MS-DOS, as well as CP/M-86. Address above.

LINK-LABEL is a unique, two-part, serially-numbered label which lets you match your diskette to its jacket. The jacket can contain descriptions of the contents of the diskette's data. The label system makes it easy to match the two, quickly and accurately, so you are sure each diskette is stored in the appropriate jacket. Contact HEXCO, Inc., P.O. Box 199-PL, Hunt, Texas 78024. A descriptive price-list/order form is free for the asking. The illustrated catalog of the full product line of specialized data processing products is $1 postpaid First Class Mail.

PARROT is a programmable speech generator for the Timex-Sinclair ZX80, ZX81, or T-S 1000, computers. Price $89.95 plus $4 handling. Contact R.I.S.T., Inc., P.O. Box 499, Ft. Hamilton Station, Brooklyn, NY 11209.

SIG/M and the CP/M Users Group have been collecting CP/M software in public domain for several years. Elliam Associates now have the complete set of disks, which includes over 2000 programs and files on over 140 disks, and are offering to copy them for you. 8" single density disks and 5-1/4" disks with over 240K are $10.00 per disk. Printed listings of the complete catalog are available for $10 each, or order diskettes CP/MUG.CAT and SIG/MUG.CAT at $10 per diskette. Formats include: IBM-PC (CP/M-86), Kaypro II, Osborne, Otrona, Superbrain, Televideo, Xerox 820, Z-disk, NorthStar, and standard 8" single density. Contact Elliam Associates, 24000 Bessemer Street, Woodland Hills, CA 91367, phone (213) 348-4278.

Disk Niche is a solid hardwood flip-top storage bin for 5-1/4" floppy diskettes. Available in solid Walnut, Oak, or Cherry, with five movable tabbed dividers, it has a capacity for 50 diskettes. "The Disk Niche offers a functional alternative in a 'plastic world'." Price $49.95 each plus $3 each for postage and handling. Visa & MasterCard accepted. Contact Systems Integration, 1519 North Nevada Avenue, Colorado Springs, CO 80907, phone (303) 635-4477. SPECIAL COMPUTER CLUB PRICE: $44.95 if you mention you heard about it through Proteus, the Processor Technology Users Group.

Macroma makes a TV Projection system that converts your ordinary TV into a projection TV that throws the screen image onto your wall or movie screen. "It's so easy a 12 year old can do it." The complete kit, including lens, is only $19.95. You supply wood or cardboard for cabinet and a common mirror obtainable locally for about $2. Kit contains precision lens system, detailed plans, complete instructions. Works with TV sets up to 26", even color sets. [Editor's note: This looks like a simple mirror-and-Fresnel-lens camera obscura projector. The image won't be super bright, but in a dark room it should work okay. Clarity will depend upon lens quality. A curved screen is required if distortion is to be avoided. The company also offers plans for a do-it-yourself curved screen.] Contact Macroma TV System, 15 N. Main Street, Washington Crossing, PA 18977, phone (215) 736-3979.

Optronics Technology makes a Disk Control Unit that turns off the motor of your 8" floppy disk drive when the disk has not been used for 9 seconds. The circuit board is the size of a business card and fits neatly into the drive, according to the manufacturer. It uses a zero-crossover control and a built-in activity monitor. It is available for Shugart 800/801/850/851, Qume DT8, Siemens and other drives. You MUST state the type of drive when you order. As a kit, the price is $29.95; assembled and tested, it is $49.95 (you install in your drive, of course). Contact Optronics Technology, P.O. Box 81, Pittsford, MY 14354, (716) 377-0369.

SOL'S CO-DESIGNER ON COVER OF INFOWORLD

Lee Felsenstein, the co-designer of Processor Technology's Sol computer, appeared as the cover story of the November 7, 1983, issue of InfoWorld, the Newsweekly for Microcomputer Users. Dubbed the "Populist Engineer", the biographical story reveals Lee's background and interests as well as his involvement in microcomputer projects, such as the "Pennywhistle modem", the Sol, the Osborne computer, and the Community Memory.

Old timers will remember the Pennywhistle modem, we all know the Sol, who can forget the Osborne, but what is Community Memory?

Community Memory embodies one of Lee's long-term dreams: decentralization of computing power; in other words, computing power for the masses. A group of counter-culture people in Berkeley and San Francisco have been working for years, apparently at their own expense, to develop an open access computerized bulleting-board system with terminals located throughout a community and available to anyone without charge.

(continued to page 14, right column)

(Meet ADA continued from page 1)

The immense momentum created by DoD spending will surely make Ada as important to computing as FORTRAN, COBOL, and yes, even BASIC, have been. And since Ada is built upon Pascal in many ways, Ada will probably replace Pascal as the teaching language in university computer science courses. Indeed, I suspect that they will actually come to like Ada.

This will take some time, though, for several reasons. One is that the systems programmers have not yet generated complete Ada compilers in any great number. In the Sept/Oct 1983 issue of The Journal of Pascal and Ada, it was reported that there were only two validated Ada compilers at that time. Many projects were working on Ada compilers for large-scale computers, but lots of bugs were still in the compilers and many were incomplete.

Another reason will come from the government's hard line on forbidding any subsets of Ada. Since the DoD has registered Ada as a trademark, it has a legal right to restrict its use. In fact, I must state that my use of the name Ada throughout this article refers to Ada (R), the "registered trademark of the U.S. Government, Department of Defense, Under Secretary for Research and Engineering," and that such use is administered by the Ada Joint Program Office (AJPO).

The DoD has adopted the policy that there will not be any approved subset or superset of Ada. Ada is Ada, and that's all there is to say about subsets or language extensions. Since Ada is a very large language, it is unlikely that standard Ada will ever be running on your little Sol, unless you enhance the Sol with a 16-bit microprocessor and more memory. And although Ada subsets with a power comparable to Pascal are quite feasible for small systems like the Sol, without an official standard every implementation will be a little different and portability will be lost. (More about this later.)

Another delay in the spread of Ada will come from the educational propogation time. There are only a few Ada experts now. It will take time to teach more. The Journal of Pascal and Ada, Vol. 2, No. 6, Nov/Dec 1983, is scheduled to have an article by Ken Bowles talking about the problems of teaching Ada. Dr. Bowles, you should remember, is the founding father of the UCSD Pascal p-system and professor of computer science at University of California, San Diego. He created a company which has been developing an Ada compiler since April 1981.

In summary, it appears to be just a matter of time before we are all hearing about Ada, and maybe even using Ada. By the way, you may wonder what the name Ada represents. Unlike FORTRAN, COBOL, and BASIC, the letters of the name Ada don't represent other words. Like Pascal, Ada was named after a real person. Ada Lovelace was the world's first computer programmer, being the lady mathematician who punched out the cards that made the Babbage computing engine do its thing.

So, Ada may be an important development, but would you want to program in Ada? Is it any good? Why bother learning it?

I asked myself these questions and decided to look into Ada a little. I've only begun to commence to initiate a preliminary overview of Ada, you might say. So, don't expect a complete tutorial here. But here are some highlights of what I've discovered in the official document defining Ada: the Reference Manual for the Ada (R) Programming Language, ANSI/MIL-STD -1815A-1983, February 17, 1983. This 3/4" thick paperback manual is available for $8.00 from the U.S. Government Printing Office.

The manual introduction states that Ada's design goals had three overriding concerns: program reliability and maintenance, programming as a human activity, and efficiency.

Program reliability and maintenance is addressed by Ada's strong typing (explained in a moment), readability, avoidance of error-prone notations, and support for modularity.

When computer scientists refer to a language as being strongly typed, they mean that a name of a programming object can only refer to one type of object, which must be explicitly defined and that types cannot be mixed without explicit conversion.

For example, in a scientific program you may be dealing with variables that have angular measurements in degrees and others with angular measurements in radians. Yet both of these variables would be declared with the same type "real". If your main program accidentally passed a degrees measurement to a subroutine where a radian parameter value was assumed, a program error would exist but the compiler would have no way to catch it. In a strongly typed language, each of these types could be declared to be distinctly different, although they would all contain real numbers as their base type. The compiler would enforce type compatibility.

A weakly typed language, on the other hand, has a few general types of objects (integer, real, character, pointer, etc.) and it forces the programmer to keep their uses straight without giving him any help. A weary programmer could get two variables confused and try assigning a time measurement in seconds (declared as a real number) to a linear variable with units of feet (also a real number), and the compiler would let it get through.

Ada avoids any such type confusion by requiring that all type conversions be explicit (unlike PL/I which has been criticized for making default type conversions with reckless abandon). That way, type errors can be caught by the compiler instead of sneaking through to become program bugs. Also, a name has only one type and that type defines not only that it represents a particular kind of data, such as an integer, but also it defines the valid range and precision of the permitted values. New instances of data types can be declared distinct from other similar types. For example,

```
HEIGHT is new DISTANCE_IN_FEET;
CIRCUMFERENCE is new DISTANCE_IN_FEET;
```

define two data types that are both based upon a previously defined type called DISTANCE_IN_FEET, but you can't pass a CIRCUMFERENCE variable where a procedure expects a HEIGHT variable. The compiler would catch this as a type error.

For example, here's a type definition from the Ada manual:

```
type COEFFICIENT is digits 10 range -1.0 .. 1.0;
```

This means that COEFFICIENT is a programmer-defined data type which can hold a real number of at least 10 significant digits and ranging from -1 to +1.

Later, you could use this type to declare other types or variables. For example:

```
subtype SHORT_COEFF is COEFFICIENT digits 5;
```

defines a constrained version of the COEFFICIENT, this subtype having only 5 significant digits. (In the computer, these may be implemented as single precision and double precision reals.) Or, another example:

```
GRID : array (1 .. 80, 1 .. 100) of COEFFICIENT;
```

defines a two-dimensional array having 80x100 elements which are all of type COEFFICIENT.

Ada resembles Pascal, but is a bit more readable. To achieve this, Ada uses quite a few reserved words, including among others:

all, and, array, at, begin, body, case, constant, declare, do, else, elsif, end, entry, exception, exit, for, goto, if, in, is, loop, null, of, or, others, out, reverse, separate, task, use, when, while, with, xor.

Variable names are made more readable by allowing the use of the underscore character as a significant character in identifiers. For example, you can define Time_Of_Day instead of TIMEOFDAY. Upper and lower case are equivalent in identifiers, and all letters are signficant.

The standard does not specify how long identfiers can be. Pascal requires identifiers to be unique in only the first 8 characters, you'll recall. Presumably there will be a limit for each Ada implementation. (It is a bit strange that there is no minimum number specified. You'd expect some minimum standard for portability, but I couldn't find it.)

Statements can have labels that are real identifiers, not just numbers. Labels are indicated by brackets made of double less than and greater than signs: <<START>>. You can:

```
goto START;
```

Literal numeric constants can be expressed in any base from 2 (binary) through 16, much like the way it is done in the C language. In Ada,

```
16#C000#
```

represents the hexidecimal number C000.

Procedure and function calls are done as in Pascal, just by naming the procedure and giving actual parameters in parentheses. But unlike Pascal, Ada allows the actual parameters to be either in the position corresponding to the formal parameter in the procedure definition, or to be identified by the name of the formal parameter it matches. For example,

```
PRINT_HEADER(HEADER => TITLE, CENTER => TRUE);
```

is a call of the procedure named PRINT_HEADER with the actual parameters being TITLE and TRUE, which respectively are to be substituted for the formal parameters HEADER and CENTER in the procedure definition. Formal parameters are the ones you give as dummy variables when you declare your procedure and give its body of statements. (Read the => symbol as "becomes".) Here it doesn't matter in what order you've placed HEADER and CENTER in the procedure parameter list. They will still be matched on the basis of the keyword.

This keyword matching of parameters is really helpful when using library routines that have lots of parameters, many of which can take default values. Ada lets you define default values and valid ranges or types of parameters. You can identify whether the parameter is an input value to the routine, an output value back from the routine, or an input/output value which is modified by the routine. You don't have to resort to error-prone calls like:

```
SORT (TABLE_A,"Table of Results",1,10,TRUE,"A",5)
```

where a couple of parameters could be reversed. You can still use position to make the correspondence between actual and formal parameters, but it is optional.

Case statements can have an "others" alternative to catch any values that didn't meet the other case alternatives. Each alternative can have several values, or expressions, or a range of values, as the selecting choices. For example,

```
case COUNTER is
  when 0 => INITIALIZE(BIN_NUMBER);
  when 1 .. 100 => UPDATE_BIN(COUNTER,BIN_NUMBER);
  when -1 | 101 => EMPTY_BINS;
  when 2*LIMIT-1 => raise NO_MORE_BINS(COUNTER);
  when others => raise  COUNTER_ERROR;
end case;
```

Arithmetic operators in Ada are similar to Pascal's, but with a few changes. The assignment operator is the same := symbol. The equal relation is uses the equals sign = but the not-equal relation uses a slash equal /= pair. There is an exponentiation operator as in FORTRAN and PL/I, the double star (**) symbol. There are no increment or decrement operators like the ones in C (+=, -=).

You can declare and initialize arrays of constants. You can declare arrays that have variables as subscript bounds (that is, you can have dynamically sized arrays). Array bounds can be passed as parameters.

Ada lets you get at the declared attributes of an identifier. For example, suppose you have a subroutine that can use a quick algorithm if it is given an actual parameter of 5 significant digits or less, but has to use a more lengthy

algorithm to satisfy the precision of a larger number of digits. You would like to check the number of "digits" that were declared for the actual parameter (see the examples COEFFICIENT and SHORT_COEFFICIENT given previously) and then branch accordingly in the procedure. Ada lets you do this. For any declared type T, the expression

```
T'DIGITS
```

yields the number of decimal digits declared. Similarly other attributes can be obtained:

```
T'LARGE yields the largest number that can fit in T;
T'EMAX yields the largest exponent value in the type;
etc.
```

Array types have special attributes that can be expressed:

```
A'FIRST yields the lower bound of the first index;
A'FIRST(N) yields the lower bound of the Nth index;
A'LAST(N) yields the higher bound of the Nth index;
A'LENGTH(N) yields the number of values in the Nth
                index range;
etc.
```

For example, if you have an array MESSAGES_TABLE that is declared MESSAGES_TABLE : array (1..MAX_MESSAGES,1..M_LENGTH), then MESSAGES_TABLE'LAST(N) has the value of MAX_MESSAGES.

Ada lets you define your own types and pass parameters to the type definition. These parameters are knows as discriminants. For example,

```
type BUFFER(SIZE:BUFFER_SIZE := 100) is
  record
    POSITION : BUFFER_SIZE := 0;
    VALUE: STRING(1 .. SIZE);
  end record;
```

defines BUFFER to be a record type having a parameter SIZE which is of type BUFFER_SIZE with default value of SIZE being 100. BUFFER contains a variable called POSITION of type BUFFER_SIZE with default value zero. BUFFER also contains a VALUE which is a string of characters numbered from one to SIZE. Later in the program, I can use BUFFER to define a variable that is an actual buffer:

```
MESSAGE_BUF : BUFFER (80) := (1..80 => ' ');
```

This declares MESSAGE_BUF to be a BUFFER of size 80, that is 80 characters should be allocated for its VALUE string. The expression after the := assignment symbol is the initial value of MESSAGE_BUF. It tells the compiler to set all 80 characters (subscripts 1 to 80) to blanks.

I hope this gives you a glimpse of the extent that Ada is like Pascal but goes beyond Pascal in giving the programmer greater power to express exactly what meanings are imposed on identifiers. Without going any further on data types, let me just say that Ada has scalar types (integer, real (fixed and floating), enumeration), array types, string types, record types (with variant parts), access types (like pointers), private types (known only inside a limited scope, as in a library routine), and task types (which permit concurrent execution of separate parts of the program).

Another feature of Ada is the pragma. A pragma is a statement which conveys information to the compiler. It is similar to Pascal's pragmatic comments, those things like:

```
(*$c+,t-,d+*)
```

which could mean to generate code, suppress the listing, and insert debugging code. These are non-standard features of many Pascal compilers. In Ada, these sorts of instructions to the compiler appear in a pragma statement. For example,

```
pragma SUPPRESS(INDEX_CHECK, ON => TABLE);
```

will suppress the compiler generated code to check that the bounds of the array TABLE are not exceeded.

Another pragma INLINE instructs the compiler to make a procedure into an "inline" routine, which is comparable to an assembly language macro in that it generates code at each call, rather than one reusable procedure.

There are 14 standard pragmas and implementers can add their own.

Another unique feature of Ada is the "overloading" of operators. In Ada-ese, this means that an ordinary arithmetic operators (such as the multiplication sign '*') can be given an additional, context-sensitive meaning. For example, if you define your own data type which you call MATRIX, you can also define the meaning of multiplying two MATRIX variables. This is done as a function definition using the operator in double quote marks as the name of the function. For example:

```
function "*" (LEFT,RIGHT : MATRIX) return MATRIX is ....
```

would be the introduction into a function definition for matrix multiplication of two matrices which are given the formal (dummy) names LEFT and RIGHT. In actual use, you would "call" this function by using the operator in its customary way:

```
A:=I*B
```

meaning that matrix A should be set to the value of matrix I multiplied by matrix B, where A, I, and B are previously declared of type MATRIX.

You can also overload relational operators. This lets you define your own ordering operation associated with a new data type.

Likewise, subprograms (procedures, functions) can be overloaded. For example, the IO packages supplied with the system define many different procedures all with the same name PUT. There is a PUT that lets you output a floating point number to a file, a PUT that sends a floating point number to a string variable instead of a file, a PUT that outputs an integer to a file, and so on. When the compiler encounters a procedure call with the name PUT it looks at the types of arguments that are being passed and compiles the code to activate the appropriate PUT procedure that matches those argument types.

The matching of parameter type profiles to pair a call with one of the overloaded subprogram definitions reminds me of the SmallTalk scheme for identifying which messages an object can understand. (See my article on SmallTalk in a prior issue.)

This overloading of operators and subprograms will let a programmer develop a very natural means of expressing operations on his own data types.

The concepts of information hiding and library units are useful in developing modular programs employing general purpose routines. In Ada, the "package" is a program unit that contains type definitions, object definitions (constants, variables), and/or subprograms (procedures, functions), some of which are hidden from the user and some which are made available to the using program.

For example, you may want to program a general purpose graphics package which has some externally known funtion names and some functions which are only needed for internal use by the code within the package.

Likewise, there may be some internal variables that are not needed outside the package. (A classic case of hidden variables is in the random number generator which keeps a secret internal pattern that it uses to generate the subsequent random number on the next call, modifying the pattern on each call.)

In Ada, a package declares which named objects are to be public and which are private. The "use" clause lets you bring into your program the definitions of names which are public (visible) in the packages you want to use.

The Ada language predefines certain library packages. The package STANDARD contains all predefined identifiers in the language, CALENDAR provides date and time routines, SYSTEM includes certain implementation-dependent characteristics (such as SYSTEM.MEMORY_SIZE), MACHINE_CODE allows insertion of native code into Ada programs, SEQUENTIAL_IO, DIRECT_IO, TEXT_IO, and LOW_LEVEL_IO are self-explanatory.

Generic units are subprograms or packages which act as a template from which corresponding actual subprograms and packages can be obtained, possibly with parameter substitution. What this says, in so many words, is that a generic unit is to Ada what a macro definition is to a macro-assembler.

The declaration of an actual subprogram based upon the generic template is called generic instantiation, and it occurs when a "new" instance of the generic unit is declared.

For example,

```
generic
  type ITEM is private;
  with function "*"(U,V: ITEM) return ITEM is <>;
function SQUARING(X : ITEM) return ITEM is
  begin
    return X*X;  -- "*" is the formal operator.
  end;
```

defines a generic function called SQUARING which takes as parameter an item X and returns the product of X with itself.

The key here is that the operator "*" is overloaded (which I previously discussed) with another meaning for ITEMs. The box "<>" is an empty declaration which takes its meaning from the context outside of the generic unit when it is instantiated.

So, if you defined a MATRIX type and its operator "*" which does matrix multiplication and a variable M of type MATRIX, then SQUARING will be able to square a matrix M. But you must first define a specific instance of SQUARING for MATRIX types, by the generic instantiation declaration:

```
function MSQUARE is new SQUARING(MATRIX);
```

Then you can use MSQUARE(M) to mean the squaring function of matrix M.

You could also declare a type COMPLEX and overload "*" with the meaning of multiplication for COMPLEX numbers. Then you could declare

```
function CSQUARE is new SQUARING(COMPLEX);
```

A commonly useful example would be a generic MIN function that takes two parameters and returns the minimum of the two. You could define this function and use new instances of it to compute the minimum of any new data type you create. For example, you may want to define a type called LOCATION which is a vector of coordinates. Then the relational operators (<,=,>, etc.) could be overloaded with definitions of your ordering of these locations (using distance from origin for example). Then instantiating the MIN function for LOCATION types would give you a new function.

Nifty? I guess so, but I think I could get along without it. In building very large, complex applications and package libraries, this kind of generic facility is helpful because it allows certain programming to be done once and then used over again when new data types are defined for other applications.

Ada also provides multi-task communication. Tasks are separate portions of the program which can execute concurrently. In the actual implementation, this may be done by time-slicing one processor or by setting multiple processors in action, each with a separate task. The tasks procede independently except at points where the programmer has arranged for them to synchronize.

Some tasks have "entries" that can be called by other tasks. A task accepts an entry call by executing an "accept" statement for the entry. This is known as a rendezvous between the task issuing the entry call and the task accepting the call. Entry calls and accept statements can have parameters, which are the principal means of communicating between tasks.

A delay statement will suspend execution of a task for a specified amount of time. A predefined library package CALENDAR supplies the type TIME and functions to read the year, month, day, and seconds. Operators "+" and "-" are defined in the package for addition and subtraction of TIME objects.

Selective waits allow the task to wait for certain conditions. Timed entry calls are cancelled if the rendezvous cannot be completed within a given delay. Conditional entry calls attempt a rendezvous but cancel if it is not immediately possible. Tasks may have priorities, may be aborted by the abort statement, and may share variables for communication.

Exceptions handlers are routines that respond to unusual conditions, such as interrupts or execution errors (zero-divide, etc.) The exceptions can be predefined (e.g., NUMERIC_ERROR) or programmer defined (e.g., SINGULAR_MATRIX). The effective range (scope) of an exception handler is indicated by where it is defined. A begin-end block (as in Pascal) may include a portion identified by the reserved word "exception". The syntax is:

```
begin
     --sequence of statements;
exception
  exception handler
  exception handler
     ...
end
```

For example,

```
begin
     COMPUTE_X
     COMPUTE_Y
exception
     when OVERFLOW => PUT (" OVERFLOW ERROR ");
     when others   => PUT (" FATAL ERROR ");
                      raise ERROR_EXIT;
end;
```

The raise statement lets the programmer voluntarily signal that a named exception has occurred and its handler should be activated. Exceptions are one way that library packages can communicate with the calling procedures. For example, the exception STATUS_ERROR is defined in the IO packages and is raised by a call to the IO routines if the file to be used is not open; the exception MODE_ERROR is raised if an attempt is made to read a file this is not an input file or write to a file that is not an output file.

And this brings up the IO packages. IO (input/output) is defined with library procedures rather than with special input/output statements. Procedures are supplied to CREATE, OPEN, CLOSE, DELETE, RESET, READ, WRITE, SET_INDEX (seek to a specified point in the file), and so on. Functions are provided to inquire on the MODE, NAME, current INDEX, SIZE, END_OF_FILE condition, and so on.

The TEXT_IO package defines routines for reading files composed as a sequence of pages, each having a sequence of lines marked by a line terminator. The actual implementation of line, page, and file terminators is not specified, but the language provides a machine independent way of detecting and creating the terminators. Routines are provided to set line and page length, create and test for terminators, read current line page and column location of the cursor, etc.

Limited formatted output, similar to Pascal's, is available, but surprisingly cumbersome. This may be a big flaw in the portability area. If every installation defines its own library routines for easy-to-use formatted IO like that in PL/I PICTURE clauses, users will have to transport the library, too.

I hope this has given you a feeling for the flavor and power of the Ada language. If you are interested in learning more about Ada, you may want to purchase the Ada reference manual I mentioned, as well as an introductory text. I'm sorry I don't yet have a reference for an introductory text, but I've read that Prentice-Hall publishes "Ada, An Advanced Introduction" by Narain Gehani, $20.25, 325 pages, which is good once you understand the basic parts of Ada.

You may also wish to subscribe to the Journal of Pascal and Ada, published every other month for $14.00 per year ($21.00 outside U.S.). Send subscription orders to Journal of Pascal and Ada, P.O. Box 384, Orem, Utah 84057.

(continued on page 8, right bottom)

(IEEE 696 Standard continued from page 1)

Digital Research has provided a new version of CP/M called CP/M 3.0, which has support for a bank selected memory scheme. I don't know much about it, but I don't see any CP/M application programs advertising that they utilize bank selected memory. If you know of some, please let me know.

MP/M-II, the multi-tasking operating system for the 8080 family of processors, also uses bank selection; but the top 16 K of each bank must not switch when banks are switched. That is, the top 16K must reside in all of the banks. Each task must have its own 48K space below, and the operating system uses the common 16K above. This could be accomplished by addressing your memory boards to have only 48K in each bank and providing one 16K board that ignores the extended lines (bank number) so it will respond to addresses in the top 16K of every bank.

Some memory boards include their own circuitry to recognize and latch bank numbers, doing their own internal bank selection without extended memory address lines. This was first introduced by Cromemco back in the early days of the S-100 bus.

What other alternatives are there? Macrotech International manufactures a dynamic memory board that can hold from 256K to 1 Megabyte in a single slot. The first 384K is on a standard sized S-100 card, and the remaining 640K is located on a piggy-back card that attaches to the host. This piggy-back makes the board thicker, but it should fit into the top slot of the Sol, where there is more space above. This board has its own built-in extended addressing latch, so that 8-bit processors can address into the 24-bit extended address space. It isn't cheap, costing $1165 for the 256K size and up to $2325 for the 1 megabyte fully populated version.

The McVideo 24x80 conversion for Sol also includes a latch with two extra bits. You could wire these to the bus (via appropriate bus drivers) so that you could have 2 bits of extended addressing (4 banks of 64K).

My feeling is this. The Sol was designed as an 8-bit microcomputer with 16 address lines. Most of the software you could want is available for the 8080 processor with 64K address space. Some software needs a Z-80 but it still runs in a 64K memory environment. So, why bother extending the Sol?

If and when you need more memory, look at the options available then and move on to either a new computer entirely or a set of S-100 boards that can reside in the 5-slot backplane of the Sol. The Sol could then be fitted with the McVideo and would become your terminal, disconnected from the S-100 bus, and just communicating through the serial port or parallel port to the system in the bus.

For example, Advanced Digital Corporation manufactures the Super Six board. This is a single S-100 IEEE 696 board with a 6 MHz Z-80, a WD2793 floppy disk controller, 2 serial ports, 2 parallel ports, DMA controller, 4K EPROM monitor, counter timer interrupt clock, and interrupt controller, plus 128K of bank selected dynamic RAM -- all on one standard S-100 board and all for $895 plus $35 for each port adapter (from Priority 1 Electronics catalog). That's a lot of computer in one slot! And it will support CP/M 3.0 in banked mode, which can be purchased already configured from Advanced Digital.

Octagon Corporation makes the Octagon 8/16 CPU which also is a standard IEEE 696 board. It has both an NSC-800 and Intel 8088 microprocessor, which gives you both Z-80 compatibility and 8088/8086 compatibility. It also has an Intel 8272 floppy disk controller with DMA into the whole 24 bit address space, 8K PROM monitor, 2 serial ports, interrupt controller, clock interrupt, and space for the 8087 numeric co-processor. It sells for $849 from Priority 1.

In summary, my recommendation is not to try re-engineering the Sol into a 24-bit address machine. Accept it as it is. Move on to a different CPU or a whole new computer when your needs are great enough.

EXTENDED I/O ADDRESSING

The 8080 processor puts I/O port addresses out onto the 16 bit address bus in what is called a "mirrored" fashion. The port address, which is only 8 bits wide for 256 ports, is repeated on the lower half and upper half of the 16 bit bus. In the Sol, the port decoder actually looks at the high order side, which is contrary to the recommendation of the IEEE standard. In the standard, the ports now address up to 64K of ports using a full 16 bit port address.

This really poses no major handicap for you. If you add new I/O interfaces that conform to the standard, you should address their ports in a way that doesn't conflict with the Sol's port addresses. As a 16 bit address, the Sol ports are F8F8, F9F9, FAFA, ..., FFFF. Just stay away from these addresses, and there will be no conflict. If you go to a 16 bit processor and disable the 8080, as suggested in Part 1 of this series and as described by Ed Bolton in this issue, you should just refer to the Sol ports by the 16 bit versions of their addresses.

New I/O boards that use consecutive 16 bit addresses may have to be modified to ignore the high-order address lines when decoding the port address if they are used in an unmodified Sol, since the Sol can't generate consecutive 16 bit addresses due to the mirroring. This can probably be done by pulling a few IC pins and tying them to ground or +5v.

TMA CONTROL

The standard allows up to 16 temporary bus masters to take over the bus from the permanent master (CPU). This allows devices such as disk controllers to temporarily use the bus to input or output data between the controller and the memory. The Helios disk controller is designed to use direct memory access, as it used to be called, but it does not use the same method for coordinating the activity of competitive controllers trying to take over the bus simultaneously.

This coordination is called bus arbitration. As we mentioned in Part 1, it uses some of the bus lines that the Sol has been using for other signals, but the signals can be relocated or abandoned. The Helios can co-exist with other DMA devices (temporary masters) as long as they are not both activated at the same time. You will have no trouble if you just be sure to let the Helios finish its entire sector transfer before you start any other TMA device going. Then there will never be a bus contention, so arbitration is irrelevant.

HELIOS DISK CONTROLLER

The Helios controller will work in a system with other TMA devices, as I mentioned above. It will not work in a system that can't supply it with a 2MHz clock, though. But, the controller, I am told, will work just fine if you have it take its clock from the IEEE 696 CLOCK signal (pin 49) instead of the master system clock Ø on pin 24.

Apparently the Helios just needs the clock for its internal timing, but the bus timing is obtained from other timing signals on the bus. I haven't tried it myself, but you could easily make the change. It just involves cutting the trace to pin 49 on the Helios controller board and jumpering that trace to pin 24 instead.

Bob Hogg tells me that this will let the Helios work just fine in a Sol with the new Z-80 upgrade installed, even when the upgrade runs at 4MHz. I am installing a Z-80 upgrade in my Sol, so I'll let you know how it goes with the Helios. If it works, you can conclude that it will work on other processors as well.

MEMORY BOARDS

Processor Tech made provisions for extended addressing on the nKRA line of memory boards (not the old 16KRA or 32KRA, but the 16KRA-1, 32KRA-1, 48KRA, and 64KRA). However, the circuitry is there in a skeletal fashion and you must add to it. In light of my earlier comments about extended addressing for the Sol, you can see why I haven't bothered to figure out the exact changes to make an nKRA into an IEEE 696 standard 24-bit address memory board.

Another handicap is that the nKRA is designed to run at system clock rates no more that about 2MHz. They won't do 4MHz. This is built into the design of the "state machine" that does the invisible refresh. So, when the time comes to move up to a bigger address space, you'll surely want to replace the memory with faster boards.

My advice: use the Processor Tech boards in the good ol' Sol and sell them or give them away when you must upgrade. Otherwise, you'll be stuck at a 2MHz system clock, and with 6, 8, and 10 MHz processors available now, who wants to cripple a speedy processor just to let it slow down for a little memory board.

WHY UPGRADE?

In summary, my feelings are this. It only makes sense to upgrade your system if you need something you don't now have. If you want to run a particular kind of software that isn't available for 8080 systems, then consider the McZOL (Z-80 upgrade) for a Z-80 microprocessor, or the trick of disconnecting the bus from the Sol motherboard and using the Sol as a terminal to your 16-bit microprocessor S-100 board system in the bus.

If you need to run software that uses more than 64K RAM, you should upgrade the processor to a 16-bit CPU. Or consider buying another computer.

Meanwhile, it still does everything it did the day you bought it.

But what to do with the Sol when you no longer are satisfied with it? My feelings are sentimental and pragmatic. Imagine what it would be like now if you had your grandfather's first car in mint condition or how about one of those old radios with the huge glowing tubes and horn-shaped speaker. It would be a classic, or at least an antique.

Go into a computer store these days. They never heard of the Sol. Not even the Intel 8080 microprocessor. It is fast becoming an antique.

Keep the Sol. Keep it in working condition. Keep all the documentation you have about it. Collect other artifacts about it -- magazine ads, articles, stories. Save them for historical purposes. Someday it will all be worth more than you paid for it.

(ADA continued from page 7)

An article in that journal has reviewed some of the Ada compilers under development. The Janus/Ada compiler by R&R Software, Inc. (P.O.Box 1512 Madison, WI 53701, 608-244-6436) runs on Z-80's under CP/M as well as the popular 16 bit operating systems for the 8088/8088. It only compiles a subset, of course, but it was commended by an expert panel at the Ada Compiler Fair for its speed on a small machine and its separate compilation feature. It costs $400 from several distributors, including Marfam Corporation, 5340 Thornwood Drive, Suite 102, San Jose, CA 95123, (408) 226-0170.

THIS 'N THAT BOARD

General Delivery
Albion B.C. Canada
V0M 1B0

Dear Stan:

When I noted that it was renewal time for the subscription to PROTEUS and saw your request for material to put into it, I thought that other SOL owners might be interested in my this 'n that board. At the time I started on it, the BSR X-10 control system was just out and Steve Ciarcia's article in Radio Electronics for an ultrasonic interface was an attractive method of connection. Since my controller wasn't an ultra-sonic controller I just added the missing components to the P.C. board and used a high frequency transformer to couple the signal from the interface. I found that his data field was slightly in error but when corrected this system worked fine. I thought that this arrangement used rather a lot of hardware for a simple function so I decided to experiment with the then new Mitel MT8804B cross point switch. This chip ties in very well with the PICO 542C controller chip used in the BSR X-10 control module. Power for the system can be drawn from the S-100 bus. The output to the line is interfaced via the output transformer from the module and two 500V disk capacitors.

Since I was interfacing this system to my SOL, I thought I would also need a real time clock to use the computer as a multi-point time controller. The NATIONAL MM58167 clock chip seemed to have all the functions one might need, and is easy to interface to the bus as well. I found that this clock chip must be put into the standby mode before the system power fails or the clock timekeeping will be altered. I added a power fail detector circuit, INTERSIL ICL 8211, to monitor the S-100 +8V rail and shut down as soon as the rail begins to drop. Unfortunately, in the SOL power system the 5V logic supply is quite independent of the S-100 bus power, and since my S-100 bus is quite lightly loaded, the main board logic fails well before the detector operates! It is necessary, therefore, to add a 10k ohm resistor from the positive on C8 in the SOL power supply to line 21 of the S-100 bus to feed the power monitor.

I also had just acquired a modem, and since the serial port on the SOL was used as a custom driver port for my printer, I needed another serial port. I decided that the widely used 8251A uart was the way to go on this port as it is more flexible than the fixed function uart's such as the AY series.

I have attached a copy of the schematic for my board, which I built up on an S-100 prototyping board using a variation on the VECTOR wiring pencil system. As is common practice on digital circuits, the layout is not too critical as long as the leads subject to interference are kept away from the data and address busses. Be sure to use bypassing, 0.1uF per chip, on the power supply lines. As can be seen, the various modules all use the common circuits (address decoding, data buffering). One can easily delete any modules not required or, indeed, add other functions, as there is lots of room left on the board.

Note that the D I/O lines from the 74LS241 buffers are connected together. The alternate pin numbers show the connections for full S-100 implimentation of the unidirectional data busses, but only those shown are needed on a SOL.

All component values are shown on the schematic except the line transformers. The H.F. transformer one is from the BSR X-10 control module as is the PICO 542C chip and their associated components. Be sure to trace the pinout of the H.F. transformer before removing it from the BSR module. The two H.F. output capacitors must be 500V ceramic for safety.

The power transformer is a reworked, cheap, "AC adaptor". Cut the case open along the joint with a thin saw (not too deep) and carefully remove the transformer and filter, noting the lead connections. Very carefully separate the core laminations and slip them out of the bobbin - heat may help if they're waxed. If there is enough room over the wrapper insulation, add a few hundred turns of #38 AWG or smaller magnet wire to the bobbin, making sure the winding does not extend beyond the lips of the bobbin. Cover the winding with one layer of hard tape eg. Magic tape. Restack the core just as you found it, being careful not to nick the new winding. Carefully test the winding insulation to be sure there is no leakage between the secondaries and the primary. You don't want to kill your SOL or yourself. Alternativly you can use two transformers such as the RADIO SHACK 273-1385 connected back to back as shown, and mounted in a mini-box. Mount the H.F. transformer and it's associated components on some vector board and mount it next to the transformer in the case. Connect the HF coupling capacitors to the line with fine wire (#38) to act as fuses. Run a length of ribbon cable from this assembly to the this 'n that board. Remember, 120V AC around the logic cards is a no-no for everybody's safety!

A part that was at first difficult to find, was the 32 kHz crystal for the clock. Go to an electronic watch repair shop for one. They are about the size of a rice grain, and though they are quite sturdy they should be added to the board last.

Attached are the lists of the access codes for each of the modules, all in decimal notation. The programs I use for the BSR and clock modules are written in Pro Tech EC Basic running under CP/M (Tad Enterprises). I have also included the source listing for the modem driver I use for accessing bulletin boards. It contains a ratrace buffer for the printer so that a 300 baud input can go to a slower printer, since most BB's have numerous pauses in their outputs to allow the printer to catch up. The printer is turned on with ↑ and off with ↓ at any time during the program. If the buffer is overfilled, the data is overwritten.

I hope this brief description of the this 'n that card and it's functions is of some use to our fellow readers, and will encourage them to submit material to PROTEUS.

Regards,

Neil Sutcliffe

See documentation on next pages...

To any member who might be able to help:

RECONE *Representação e Consultoria Empresarial S/C. Ltda.*

Re: Extended Disk BASIC (Latest Vs.)

We are users of the SOL Computer and would like to obtain, at your earliest convenience, the latest revision of

Extended Disk BASIC
14 digit precision (2 sets)
16 digit precision (2 sets)

Please contact our New York representative, Mr. LUIZ F. BORGES, c/o INTERPROM Inc. - 60 E, 42nd St., Rm.4511, New York, N.Y. - 10165, and make payment and delivery arrangements with him.

("This and That" Board, continued)

CONTROL CODES FOR MM58167 CLOCK CHIP

ADDRESSES		FORMAT OF I/O
0	MILLISEC	
1	1/100,1/10 SEC	D0 - D3 = UNITS
2	SECONDS	D4 - D7 = TENS
3	MIN	
4 COUNTER	HOUR	UNITS NOT USED ON MILLISEC
5	DAY OF WEEK	TENS PARTIALLY USED ON ALL
6	DAY OF MONTH	BUT MILLISEC AND 1/10 SEC
7	MONTH	
8	MILLISEC	COUNTER AND LATCH RESET BITS
9	1/100,1/10 SEC	
10	SECONDS	BIT DECIMAL FUNCTION
11	MIN	0 1 MILLISEC
12 LATCHES	HOUR	1 2 1/100,1/10 SEC
13	DAY OF WEEK	2 4 SEC
14	DAY OF MONTH	3 8 MIN
15	MONTH	4 16 HOUR
16 INTERRUPT STAT REG (READ)		5 32 DAY OF WEEK
17 INTERRUPT CONTROL REG (WRITE)		6 64 DAY OF MONTH
18 COUNTER RESET		7 128 MONTH
19 LATCH RESET		
20 STATUS BIT		NOTE: LATCHES SET WITH 1 IN TWO
21 GO COMMAND		MSB'S OF THE BYTE CAUSE A 'DON'T
22 STANDBY INTERRUPT		CARE' STATE IN THE LATCH.
23 NON FUNCTIONAL		STATUS BIT: A 1 IN D0 INDICATES A
24 "		ROLL OVER DURING THE LAST READ OF
25 "		OF A COUNTER. D1-D7 = 0
26 "		GO: A GO COMMAND RESETS 1/1000,
27 "		1/100,1/10 AND SEC THEN STARTS
28 "		THE CLOCK.
29 "		STBY INTERRUPT: A 1 IN D0 SETS THE
30 "		INTERRUPT; A 0 DISABLES IT. THIS
31 TEST MODE (FACTORY)		OPERATES DURING POWER DOWN.

CONTROL CODES FOR BSR-X10 CONTROLLER

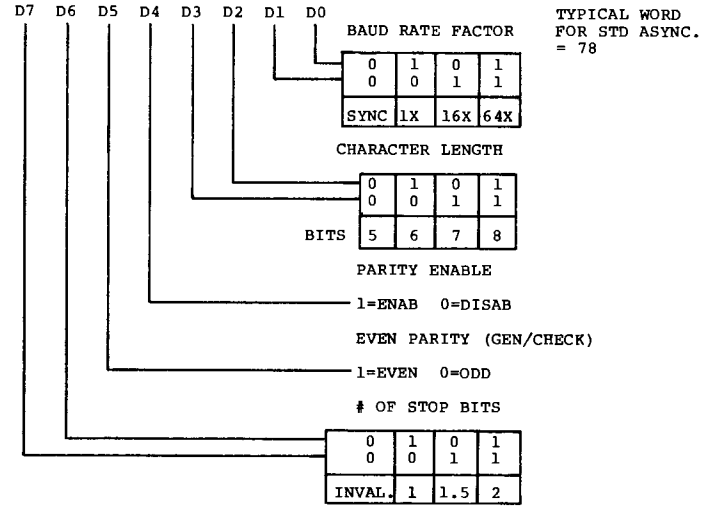
PORT # = 64

FUNCTION	CODE	FUNCTION	CODE
STA# 1	22	STA# 13	18
2	38	14	34
3	19	15	16
4	35	16	32
5	21	ON	67
6	37	OFF	65
7	20	ALL ON	69
8	36	ALL OFF	66
9	23	BRIGHT	68
10	39	DIM	64
11	17	RESET	08
12	33		

CONTROL CODES FOR 8251A UART

ADDRESS: DATA = 96
MODE/CONTROL = 97
} UNDECODED ADDRESS FOR 32 PORTS

MODE WORD: 1st CONTROL WORD AFTER A RESET (WRITE)
N.B. HARDWARE RESET IS NOT IMPLIMENTED - USE SOFT RST.



CONTROL WORD: (WRITE)

D0 - Tx ENABLE	1=EN	0=DIS
D1 - DTR	1=DTR+	0=DTR-
D2 - Rx ENABLE	1=EN	0=DIS
D3 - SND BREAK	1=BRK	0=NORM
D4 - ER	1=RESET	0=NORM (ERROR RESET PE, OE, FE)
D5 - RTS	1=RTS+	0=RTS-
D6 - IR(RESET)	1=RESET	0=NORM
D7 - EH	1=SEARCH FOR SYNC CHARACTER (ENTER HUNT MODE)	

STATUS: (READ)

D0 - Tx RDY	1=DATA BUFFER EMPTY	
D1 - Rx RDY	1=CHR READY IN RECEIVER	
D2 - Tx EMPTY	1=TRANSMITTER EMPTY	
D3 - PARITY ERR	1=ERROR	} RESET WITH 'ER'
D4 - OVER RUN	1=ERROR (CHR NOT READ BEFORE NEXT RCVD.)	
D5 - FRAMING	1=ERROR (NO VALID STOP BIT)	
D6 - SYNDET	1=SYNC CHR (SYNC MODE ONLY)	
D7 - DSR	1=DSR INPUT=+ 0=DSR INPUT=-	

("This and That" Board Schematics)

FIG.1

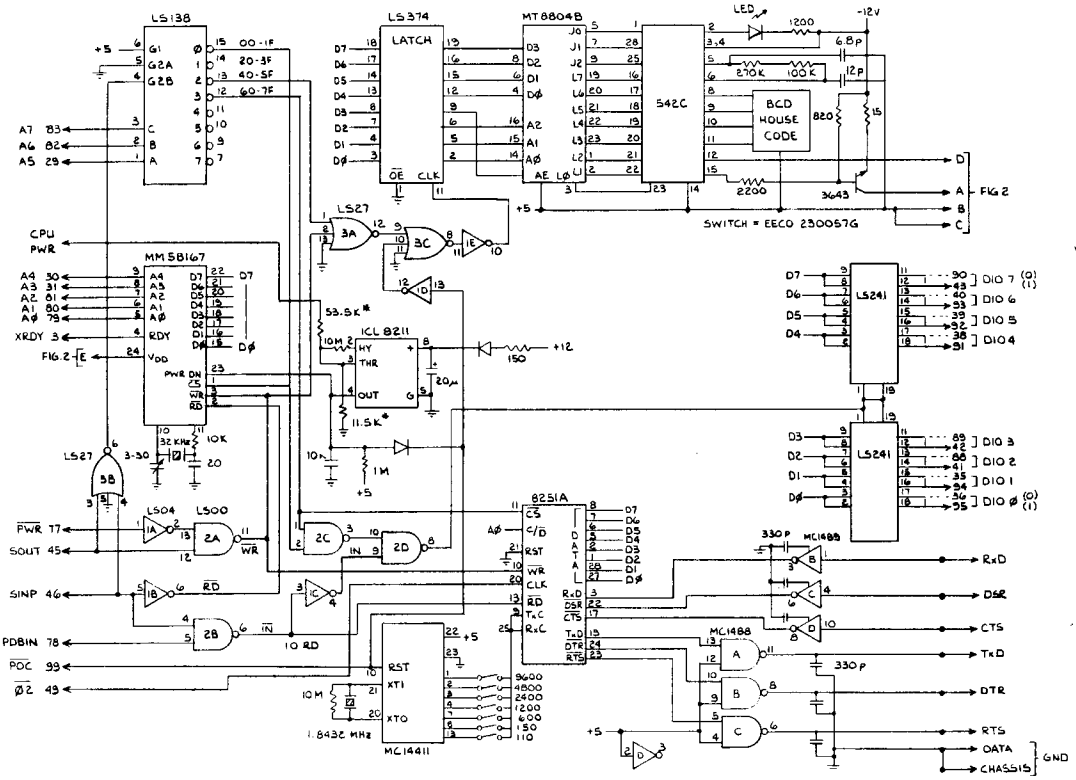
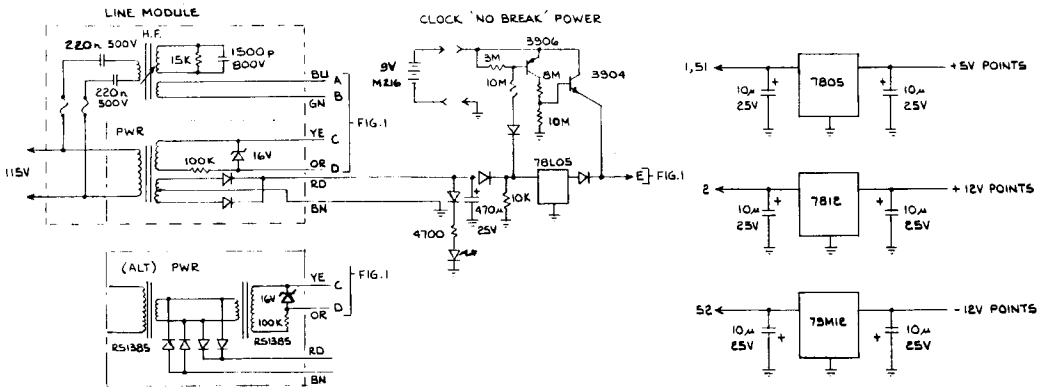


FIG.2



POWER PINS

	LS00 LS04 LS27	LS138	LS374 LS241	8251	8804	14411	1488	1489
+12							14	
+5	14	16	20	26	24	24		14
GND	7	8	10	4	12	12	7	7
-12					13		1	

(This and That, listing)

LI.

```
 10 REM; TEST AND DIRECT CONTROL PROGRAM FOR BSR SECTION OF
 20 REM; THIS N' THAT CARD
 30 INPUT "STATION ",S
 40 IF S=0 THEN  GOTO 220
 50 ON S GOTO 60,70,80,90,100,110,120,130,140,150,160,170,180,190,200,210
 60 LET O=12: GOTO 310
 70 LET O=28: GOTO 310
 80 LET O=4: GOTO 310
 90 LET O=20: GOTO 310
100 LET O=2: GOTO 310
110 LET O=18: GOTO 310
120 LET O=10: GOTO 310
130 LET O=26: GOTO 310
140 LET O=14: GOTO 310
150 LET O=30: GOTO 310
160 LET O=6: GOTO 310
170 LET O=22: GOTO 310
180 LET O=0: GOTO 310
190 LET O=16: GOTO 310
200 LET O=8: GOTO 310
210 LET O=24: GOTO 310
220 INPUT "FUNCTION ",F$
230 IF F$="ALL ON" THEN  LET O=3: GOTO 310
240 IF F$="ALL OFF" THEN  LET O=1: GOTO 310
250 IF F$="ON" THEN  LET O=5: GOTO 310
260 IF F$="OFF" THEN  LET O=7: GOTO 310
270 IF F$="DIM" THEN  LET O=9: LET S=20: GOTO 310
280 IF F$="BRIGHT" THEN  LET O=11: LET S=20: GOTO 310
290 IF F$="NEW" THEN 30
300 PRINT "NOT VALID FUNCTION": GOTO 220
310 OUT 253,O
320 PAUSE 10
330 OUT 253,128
340 IF S=0 THEN  GOTO 220
350 IF S=20 THEN  GOTO 370
360 GOTO 30
370 INPUT "MORE?",D$
380 IF D$="Y" THEN  GOTO 310
390 IF D$="" THEN  GOTO 310
400 LET S=0: GOTO 220
410 END
```

LI.

```
 10 REM; BSR 100 EVENT TIME OF DAY CONTROL IN PTC EXTENDED BASIC
 20 REM; INPUTS MUST BE IN CHRONILOGICAL ORDER
 30 DIM C(20),S(20),A(100),B(100),D(100),E(100)
 40 REM; TRANSLATION TABLE FROM POINT NUMBER TO CONTROL CODE
 50 DATA 12,28,4,20,2,18,10,26,14,30,6,22,0,16,8,24
 60 FOR X=1 TO 16
 70   READ C(X)
 80 NEXT X
 90 FOR L=1 TO 100
100   LET B=7: REM. OFF CODE
110   INPUT (2,0)"UNIT  ",A
120   LET A(L)=C(A): REM; A(L)=CONTROL CODE FOR POINT
130   INPUT (2,0)"FUNCTION  ",B$
140   IF B$="ON" THEN  LET B=5: REM. ON CODE
150   LET B(L)=B: REM; B(L)= ON OR OFF CODE
160   INPUT (2,0)"HOURS  ",C: GOSUB 240
170   LET E(L)=Q: REM; E(L)= HOURS IN BCD
180   INPUT (2,0)"MINUTE  ",C: GOSUB 240
190   LET D(L)=Q: REM; D(L)= MINUTES IN BCD
200   INPUT (1,0)"MORE ?",A$
210   IF A$="N" THEN  LET P=L: EXIT 300
220 NEXT L
230 REM; DECIMAL TO BCD CONVERSION
240 LET X=INT(C/10): REM; FIND 10'S
250 IF X=0 THEN  LET Q=C: RETURN
260 LET Y=16*X: REM; MULTIPLY 10'S BY 16
270 LET Z=C-10*X: REM; STRIP 10'S FROM C
280 LET Q=Y+Z: REM; FORM TWO NIBBLE BCD VALUE
290 RETURN
300 FOR R=1 TO P
310   REM.CLOCK READ AND CONTROL OUT LOOP
320   IF INP(4)<E(R) THEN  GOTO 320: REM; TEST FOR HOUR MATCH
330   IF INP(4)=E(R) THEN  GOTO 350
340   IF INP(4)>E(R) THEN  GOTO 420
350   IF INP(3)<D(R) THEN  GOTO 350: REM; TEST FOR MINUTE MATCH
360   IF INP(3)=D(R) THEN  GOTO 400
370   IF INP(3)>D(R) THEN  GOTO 420
380   REM; DO OUTPUT ROUTINE TO BSR CHIP WITH APPROPRIATE CLEARS
390   REM; AND WAITS FOR THE MODULES TO REACT.
400   OUT 253,A(R): PAUSE 10: OUT 253,128: PAUSE 2
410   OUT 253,B(R): PAUSE 10: OUT 253,128: PAUSE 2
420 NEXT R
430 END
```

LI.

```
 10 REM; CLOCK READ ROUTINE
 20 CURSOR 1,28
 30 FOR R=7 TO 2 STEP -1
 40   LET N=INP(R)
 50   LET X=INT(N/16)
 60   IF X=0 THEN  LET M=0: GOTO 80
 70   LET M=X
 80   LET L=N-X*16
 90   ON R GOSUB 600,600,500,500,400,350,200
100 NEXT R
110 PAUSE 9: GOTO 20
200 ON (M*10+L) GOSUB 220,230,240,250,260,270,280,290,300,310,320,330,340
210 RETURN
220 PRINT "JANUARY   ";: RETURN
230 PRINT "FEBRUARY  ";: RETURN
240 PRINT "MARCH     ";: RETURN
250 PRINT "APRIL     ";: RETURN
260 PRINT "MAY       ";: RETURN
270 PRINT "JUNE      ";: RETURN
280 PRINT "JULY      ";: RETURN
290 PRINT "AUGUST    ";: RETURN
300 PRINT "SEPTEMBER ";: RETURN
310 PRINT "OCTOBER   ";: RETURN
320 PRINT "NOVEMBER  ";: RETURN
330 PRINT "DECEMBER  ";: RETURN
340 PRINT "DECEMBER  ";: RETURN
350 IF M=0 THEN  PRINT " ";
360 PRINT M*10+L;" ";: RETURN
400 ON (M*10+L) GOSUB 420,430,440,450,460,470,480
410 RETURN
420 PRINT "SUNDAY    ";: RETURN
430 PRINT "MONDAY    ";: RETURN
440 PRINT "TUESDAY   ";: RETURN
450 PRINT "WEDNESDAY ";: RETURN
460 PRINT "THURSDAY  ";: RETURN
470 PRINT "FRIDAY    ";: RETURN
480 PRINT "SATURDAY  ";: RETURN
500 IF M=0 THEN  PRINT " ";
510 PRINT (M*10+L);":";: RETURN
600 IF M=0 THEN  PRINT " ";
610 PRINT (M*10+L);: RETURN
```

LI.

```
 10 REM; SET CLOCK ROUTINE
 20 FOR D=7 TO 3 STEP -1
 30   ON D GOSUB 200,200,240,230,220,210,200
 50   LET X=INT(N/10)
 60   IF X=0 THEN  LET Q=N: GOTO 100
 70   LET Y=16*X
 80   LET Z=N-10*X
 90   LET Q=Y+Z
100   OUT D,Q
110 NEXT D
120 INPUT (1,0)"INPUT 0 TO START CLOCK",Q
130 OUT 21,Q
140 END
```

```
200 INPUT (2,0)"MONTH ",N: RETURN
210 INPUT (2,0)"DATE ",N: RETURN
220 INPUT (1,0)"DAY OF WEEK (SUNDAY = 1 ) ",N: RETURN
230 INPUT (2,0)"HOUR ",N: RETURN
240 INPUT (2,0)"MINUTE ",N: RETURN
250 INPUT "",N: RETURN
```

```
;MODEM AND PRINTER CONTROL PROGRAM FOR 8251A
;BY N.SUTCLIFFE
;ISSUE 3 81/08/20
;THIS PROGRAM IS LOADED AT 100H TO RUN A
;MODEM THROUGH THE THIS N' THAT CARD
;AND PRINTER THROUGH PORT #3
;
ADUT    EQU     0C01CH
CTLW    EQU     97H
CTLZ    EQU     9AH
KSTAT   EQU     0C02EH  ;KBD STATUS
MODE    EQU     80H
VDMOT   EQU     0C054H
CR      EQU     0DH
LF      EQU     0AH
ESCFL   EQU     0C80CH
TOP     EQU     0AH     ;OVER THE TOP FLAG
BASE    EQU     500H    ;START DATA HERE
SERST   EQU     0F8H    ;PTR STAT PORT
SCTS    EQU     20H     ;STAT MASK
SDROT   EQU     0C04AH  ;SERIAL PORT
ESC     EQU     1BH
;
        ORG     100H
MODEM:  MVI     A,40H   ;RESET CODE
        OUT     61H     ;TO UART
        MVI     A,4FH   ;MODE WORD
        OUT     61H     ;TO UART
        MVI     A,27H   ;CONTROL WORD
        OUT     61H     ;TO UART
        LXI     H,0C815H
        MOV     M,00
        LXI     H,BASE  ;SETUP DATA SPACE
        SHLD    PNTS
        SHLD    PNTP
;
MODEM1: CALL    KSTAT   ;LET'S GO
        JZ      TIN
        MOV     B,A
        CPI     MODE
        JZ      CPM
        CPI     CTLW
        JZ      PRN1
        CPI     CTLZ
        JZ      NPRN1
        JC      MOUT1   ;OUT TO PORT
        CALL    VDMOT
        JMP     TIN
PRN1:   CALL    PRN
        JMP     TIN
;
NPRN1:  CALL    NPRN
        JMP     TIN
;
;
MOUT1:  CALL    MOUT    ;SEND TO MODEM
;
TIN:    CALL    PRINT   ;SEE IF THERE IS ANYTHING
        CALL    SMOD    ;MODEM STATUS
        JZ      MODEM1
        ANI     7FH     ;NO HIGH BITS
        JZ      MODEM1
        MOV     B,A     ;IT'S OUT FROM B
        CPI     1BH
        JNC     TERM2
        CPI     CR
        JZ      TERM2
        CPI     LF
        JZ      TERM2
        LDA     ESCFL
        ORA     A
        JNZ     TERM2
        PUSH    B
        MVI     B,ESC
        CALL    VDMOT
        MVI     B,7
        CALL    VDMOT
        POP     B
;
;
;
TERM2   EQU     $
        CALL    PTR
        CALL    VDMOT
        JMP     MODEM1
;
;
MOUT:   IN      61H
        ANI     01H     ;STRIP STATUS FOR TX BUFFER
        JZ      MOUT
        MOV     A,B     ;PUT DATA IN A
        OUT     60H     ;GIVE IT TO UART
        RET             ;DONE
;
;
SMOD:   IN      61H     ;UART STATUS
        ANI     02H     ;RX STATUS BIT
        RZ              ;CARRY ON IF NONE
        IN      60H     ;GET BYTE
        RET             ;USE IT
;
;
PRN:    PUSH    H       ;SET UP TO PRINT
        LXI     H,0C815H;CONTROL ADDRESS
        MVI     M,03    ;ON FLAG AND DEVICE CODE
        POP     H
        RET
;
;
NPRN:   PUSH    H       ;AS ABOVE BUT OFF
        LXI     H,0C815H
        MVI     M,00
        POP     H
        RET
;
;
PTR:    PUSH    H
        LXI     H,0C815H
        MOV     A,M

        POP     H
        CPI     00
        RZ
        CALL    SAVE
        RET
;
CPM:    RET
SAVE:   PUSH    H       ;SAVE H,L
        LHLD    PNTS    ;GET POINTER
        MOV     M,B     ;STORE CHAR FROM B
        INX     H       ;STEP H,L
```

```
132:            MOV     A,H     ;GET HIGH
133:            CPI     TOP     ;IS IT OVER?
134:            JNZ     DONE
135:            LXI     H,BASE  ;START OVER
136: DONE:      SHLD    PNTS    ;SAVE ADDRESS POINTER
137:            POP     H
138:            RET
139: ;
140: ;
141: PRINT:     IN      SERST   ;GET UART STAT
142:            ANI     SCTS    ;TEST IT
143:            RZ              ;IT'S BUSY DON'T WAIT
144:            PUSH    H       ;SAVE H,L
145:            PUSH    D       ;SAVE D,E
146:            LHLD    PNTP    ;GET POINTER
147:            MOV     A,H     ;GET HIGH
148:            CPI     TOP     ;IS IT OVER
149:            JNZ     GOOD
150:            LXI     H,BASE  ;START OVER
151: GOOD:      XCHG            ;SWAP H-L,D-E
152:            LHLD    PNTS    ;GET SAVE POINTER
153:            MOV     A,E     ;LOW ADDRESS SAME?
154:            CMP     L       ;
155:            JNZ     PNT     ;NO? PRINT THEN
156:            MOV     A,D     ;HIGH ADDRESS SAME?
157:            CMP     H       ;
158:            JZ      FIN     ;IF SO THERE IS NONE LEFT
159: PNT:       XCHG            ;GET THE POINTER BACK
160:            MOV     B,M     ;GET THE BYTE
161:            INX     H       ;STEP POINTER
162:            SHLD    PNTP    ;SAVE THE POINTER
163:            CALL    SDROT   ;PRINT IT
164: FIN:       POP     D       ;PUT IT BACK
165:            POP     H
166:            RET
167: ;
168: PNTP:     DS      2       ;PRINTER POINTER STORE
169: PNTS:     DS      2       ;SAVE POINTER STORE
170: ;
171: ;
172: END
173:
174:
```

(End of "This and That")

FOR IMMEDIATE RELEASE

APRIL 15, 1983

OSRA

OFFICE SYSTEMS RESEARCH ASSOCIATION

OSRA Issues Call for Papers
for 1984 Research Conference

Contacts: Joel D. Levy, Conference Chairman
115 Maywood Drive
Rochester, NY 14618
Phone: (716) 461-2171
275-4256

Edward G. Thomas, OSRA President
Cleveland State University
Cleveland, OH 44115
Phone: (216) 687-4740

Cleveland (April 15, 1983)--The Office Systems Research Association (OSRA) has issued a Call for Papers for its Third Annual Research Conference scheduled for February 18-19, 1984 in Los Angeles. The conference will immediately precede the AFIPS Office Automation Conference which is scheduled for February 20-22, 1984 in Los Angeles.

(SOL's Co-Designer continued from page 3)

Record stores, supermarkets, libraries, etc. would have terminals all tied to a central computer in the city. People would be able to enter any sort of message they want, just as they now can place a note upon a common bulletin board.

"Community Memory's ultimate role," the story states, "may be to pull together all of the bulletin boards and resources of the country into one network that is accessible to everyone."

It was through the Community Memory group that Felsenstein and Bob Marsh, the other co-designer of the Sol, met and began to collaborate on computer add-ons for the MITS Altair computer. Marsh and another friend formed the Processor Technology Corporation, contracted with Felsenstein for the design work, and the 4KRA memory board, the CUTS tape interface, and VDM video display were born.

Late in 1975, Les Solomon, the technical editor of Popular Electronics (now called Computers & Electronics) magazine, challenged Bob Marsh to come up with a low-cost design for a computer terminal within one month and promised to make it a cover story on his magazine. This fit into one of Lee Felsenstein's earlier goals, namely to design a low-cost "Tom Swift Terminal" for Community Memory. So, working days and nights, Marsh and Felsenstein designed and built the first Sol. The rest of the Sol story we know all too well.

What's Lee Felsenstein doing these days? He still moderates the Homebrew Computer Club, which was the breeding ground for many microcomputer companies and where I got my first introduction to the Sol.

I recall the night when Steve Jobs came to the Homebrew meeting with the first production version of the Apple II case and held it up in the air for all to see. Everyone was quite impressed that these two kids, Wozniak and Jobs, had gotten together the big bucks to have a real structural-foam case designed and built. Remember, in those days of 1976, microcomputers all had low-technology, sheet-metal and screws, cases which were cheap to make in small quantities. The tooling for the foam cases cost a bundle, but would pay off when production was huge. No one dreamed how huge it would become. Apple has sold more than a million Apple II's, and the end is not in sight.

And Lee is also working on another of his long-term ideas through his company called "Golemics". The name draws from the ancient Hebrew tale about a "golem", a creature made of clay and brought to life by religious rituals, but which got into a water-fetching infinite loop and flooded the entire village. Of course, today we call these things robots.

Unlike a robot, Felsenstein's idea of a golem would be a machine which has some autonomous features of a robot but also depends upon a close interaction with a human operator. Golemics is what the golem lacked. "...robotics keeps the user outside the feeback loop, while golemics brings the user inside the loop. Unlike the original golem, the modern version can't run wild anymore than a hammer can run wild."

I hope these highlights of the InfoWorld story will encourage you to find a copy of this issue and read more about Lee's life and aspirations. He's an interesting guy, and I'm glad that InfoWorld is bringing out some of the human side of this high-technology era.

His story reminds me of the frontier men in the early days of settling into this continent. These loners would go out for long times into the beautiful but wild, uncharted country, sometimes returning to work for one of the civilized entrepeneurs who needed their knowledge of the territory, yet always going back out beyond the frontier following their dreams.

I'm sure we haven't heard the last about Lee. He seems to have a great resiliency and bounces back into the visible universe every once in a while from the fringes of his frontier.

SAY, WHAT'S BOB MARSH UP TO THESE DAYS?

The story about Lee made me realize that perhaps you would like to know what happened to the other people involved in Processor Technology. I'll look into the others for a later issue, but I can tell you that Bob Marsh is one of the founders of a company that manufactures a RAM disk add-on for the Osborne computer, known as "Drive C." This is a little box that plugs right into the Osborne I's diskette storage slot, and becomes a super-fast disk emulator that CP/M can use as drive C.

The demise of Osborne Computer Corporation put a brake on their efforts to hitch onto the speeding Osborne locomotive of success, but there still are lots of Osborne computers out there. Xerox is providing service for Osbornes, so the computers will not be junked yet, and the add-on market may provide a steady but small income to those who have specialty products for Osbornes.

Drive C was reviewed in InfoWorld recently. I can't cite the issue because I've misplaced mine, but it is in November 1983, I believe.

...ON A DATA BASE MANAGER AND ELECTRONIC SPREAD SHEET

Dear Stan,

I would like to have the use of a data base manager and an electronic spread sheet. I have a Sol/Helios system. Could you tell me what I need to do to get this on my Sol? I am trying to convert the spread sheet program from the last Proteus but am having some dificulty. It seems that I need to have a character in the string variable in order to specify particular parts of the string to point to (ie. A$(2,4) must have a length of at least 4 characters or else I get an OUT Of BOUNDS error). This becomes a problem with a Data Base program I am trying to work out also.

Sincerely,

Roy Heybrock

Roy Heybrock
2516 Lakeshore Drive
Greensboro, NC 27407

...ON AMATEUR RADIO SOFTWARE FOR SOL

October 17, 1983

Francis R. Biasiotto
135 West 31st Street
New York City, NY 10001
(212) 736-8500

Dear 'Proteus':

Can anyone in your organization steer me toward any SOL programs suitable for Amateur Radio work. My SOL has 48K of memory and 2 drives (HELIOS) also tape.

Enclosed is a copy of a letter from Joe A. Elliott of the Flesher Corporation. His outfit makes the hardware necessary to connect the SOL to the ham radio set. If you can help me with the software I'll have all the ingredients.

Any assistance you can lend will be very much appreciated.

PTC BASIC RISES AGAIN AS "NEVADA BASIC"

Processor Technology's Extended Disk BASIC with Business Basic features, has been enhanced and adapted to CP/M by Ellis Computing, Inc. If you are looking for a transition from your Extended Cassette or Extended Disk BASIC programs to a CP/M compatible BASIC, this is the way to go. See the press releases that follow in this issue. (Chuck Ellis is a Proteus member and has been licensed for commercial use of the PTC BASIC and PTDOS EDIT.) PTC FORTRAN is also available for CP/M from Ellis, under the name "NEVADA FORTRAN".

[Editor's reply: For a real, commercially produced database manager and spread-sheet program, I recommended getting CP/M for your Helios and buying these programs from Lifeboat Associates. For a do-it-yourself, simple spread-sheet, we now have Hirner's ESS program converted to Helios/EDBASIC. It was donated to the Helios library by Leonard Cole, and will be described in the next issue. --Stan.]

(Press Release)

ELLIS COMPUTING, INC.
SOFTWARE TECHNOLOGY

3917 Noriega Street San Francisco, CA 94122 U.S.A. (415) 753-0186

COMMODORE-64 GETS COBOL AND FORTRAN

August 16, 1983

Ellis Computing, Inc., has signed two agreements with Commodore Business Machines, Inc., to provide Nevada COBOL and Nevada FORTRAN under the CP/M operating system for the Commodore-64.

Both Compilers will be made available worldwide, exclusively through Commodore's standard distribution channels. Production is well under way, with deliveries set for early this fall.

Ellis Computing, Inc., is a San Francisco based Software Development Firm located at 3917 Noriega Street, San Francisco, Ca., 94122.

...ON PT BOARD DOCUMENTATION

Sept. 9, 1983

Ed Rainsberger
4715 Kentwood Lane
Dale City, VA 22193
703-590-5068 after
3 PM your time

Dear P. T. Users:

I have six P. T. Boards that I need documentation for. They are bare boards: one is a CPU board #PC110201 Rev B, a 5 slot backplane, a GPM, a CUTS and a 16K to 64K DRAM, and a 2708 personality module. I would like to know if any PROTEUS members would have any circuit drawings and parts lists for them. I am willing to trade the CPU & personality module for documentation on the S100 boards, or if no one wants to trade, I am willing to pay the cost of copying and postage.

...ON DISC DRIVES OTHER THAN HELIOS

STATE UNIVERSITY OF NEW YORK
DOWNSTATE MEDICAL CENTER
• DEPARTMENT OF MEDICINE
MAILING ADDRESS: 450 CLARKSON AVENUE, BROOKLYN, NEW YORK 11203 / PHONE: (212) 270-1000

Dear Stan:

Please find my check for thirty dollars ($30.00) for renewal on my subscription to Proteus.

At this time, I would like to learn how the Sol can be loaded into Basic and kept that way without having to load with the casette tape each time I want to use it.

Can you also give me more information on how disc drives, other than the Helios, can be used with the Sol. This information would be very helpful to a novice.

Thank you.

Sincerely,

Harold A. Lyons

Harold A. Lyons, M.D.
Professor of Medicine
Director, Pulmonary
Disease Division

Editor's reply:
BASIC can be written into an EPROM (Erasable Read-Only Memory) board, but a better answer is to buy a disk system. It allows rapid loading of programs and opens a whole world of power you'll never have in cassette systems. See next letter....

...ON QUESTIONS ABOUT DISK CONTROLLERS

Dear Stan,

Thankyou for your response to my inquiry as to which disk controller is most recommended. As a result, I now have more questions. Please bear with me as I know next to nothing about disk drive and DOS compatibility, and want to make my first purchase the right one.

[Editorial reply is handwritten.]

1) Which disk drive controller is most recommended and why (5 1/4 or 8 inch)? 8" is more standard for software interchange, but most people now buy 5¼". There's no clear standard 5¼" format. See my note next column.
2) I might eventually want a second drive. Which controller could handle two? As far as I know, all will handle at least two drives.
3) What disk drive is recommended? Tandon and Shugart are leaders, but others are okay, too.

(continued to bottom right)

...ON A REAL TIME CLOCK INTERFACE

Dear Sir,

I have been trying to interface National Semiconductor's 58174A Real Time Clock to the SOL and have been having trouble with it. I believe timing is the main problem and wait states need to be generated, but so far all attempts to do this have met with failure. Has anybody that you know of managed to design an interface for this chip to the SOL, and would they be willing to let me have the circuit involved, as I am most anxious to get the clock working. I have written the software to display the time and date on listings and did have an MM58174 that worked for a while until the ambient temperature rose, so I was able to get the software working, and would be willing to swap it for the appropriate hardware. I have the clock wired as part of memory at the moment, and would prefer to use it this way rather than have to use input/output ports.

I hope you can be of assistance in these matters as there is no SOL User's Group over here, and most people haven't even heard of the SOL.
Thanking you in anticipation,
Yours faithfully,

Box 127,
Woodville.5011.
Sth.Aust.
26th March 1983.

Bruce D.Bull

A Note about disk formats: Before IBM produced the IBM PC, there was no one format for 5¼" diskettes that emerged as an overwhelming standard. Since the IBM PC and its copies have such a huge share of the market now, one could say that the IBM PC format is the standard. Softech Microsystems (the UCSD P-System vendor) calls it "The Universal Medium." — Stan Sokolow

(continued from bottom left)

4) Which format is most useful/cost efficient and has best software support by users or manufacturers? It's hard to say. NorthStar is popular 5¼". See my reply to Slade's letter, p.19.
5) Which format has Basic available from a reliable source? Most, if not all.
6) Does such basic allow usage of cassette Basic programs? Nevada BASIC (see left column) is very similar to Proc.Tech cassette BASIC.
7) Is CP/M available for both formats? Yes. Usually from the vendor who sells the controller board.
8) To what extent must the DOS be tailored to the various parts of the system? What features are there? CP/M must be customized for your specific hardware. You'll need expert help.
9) Which memory boards are recommended / most compatible for all uses in Sol? Static memory has more compatibility because it isn't as dependent upon bus cycle timing as dynamic memory.
10) I purchased an IMSAI mainframe to expand Sol's buss. Is there any special confisuration for the S 100 extension that I'm going to construct? See my article on the S-100 bus standard in last issue. Also see back issues, A few people have done this.
11) Which video board would give best results/dollar with little or no modifications to Sol? That depends upon what features you want: high resolution graphics, color, etc. For simply 24x80 characters, our MicVideo is best for Sol.
12) What disk hardware/software addition to Sol is most recommended? CP/M*24x80 video.

Perhaps a reader would like to elaborate on my answers. Space doesn't permit more detail here.

C. Jason Slade,
32 Berkley Cres.,
Simcoe, Ontario.
N3Y 2K4

...ON ECB FILES WITH G/2 BASIC

Stan Sololow,
PROTEUS/News,
1690 Woodside Road, Suite 219,
Redwood City, California 94061-3483
U. S. A.

John D. Barber
28 Bellevue Avenue,
London, Ontario, Canada.
N6C 4A6
February 17, 1893

Dear Stan,

Enclosed is a photostat of a photostat of a manual of a graphics upgrade for the SOL/VDM. I am sorry that the quality is not better. I do not know if there is a copywrite, or who would own it. I found it in the back of the manual when I bought another SOL. It might be of use to someone.

Also enclosed is $24. I hope it is enough dues for next year. You never seem to bill me in time. I will send more if you raise the price.

For those who have G/2 BASIC, this may be of some use. If you plug two SOLs together as described in the Chess-80 manual, and load G/2 BASIC into one, and Extended Casette BASIC into the other, you may translate files to a format that can be read by ECB. Set the baud rate as slow as possible, 75 baud, and the rest of the serial port switches the same. In the SOL with ECB type SET IP=1. In the other SOL type RENUM 100,10, SET O=1, and then LIST 100-. The SOL with ECB will treat the SOL with G/2 BASIC as a fast typist. You will also lose the ends of some long lines, and the line numbers of the following lines, as the transfer rate is still too fast for BASIC. Try to keep track of those lines and list them separately later. Erase the lines below 100. They will be spurious copies of higher numbers.

It is a very slow and tedious job, as well as being not very rewarding, so make sure the program is worth the effort.

For a Christmas wish list. How about the three upgrades from Mr. Hogg on a single new PC board? Their total price is about $500 and it seems that the three together could be made for no more than that. I have McVideo and very much like it, but I am not sure I trust the piggyback aspect. I rather think that a Z-800 rather than a Z-80 should do the job. Does PROTEUS own the copyright for the SOL pcb? Is it in the public domain? Such a mod would probably void the copyright in any case. It would be very extensive.

Keep up the good work. Thanks for the last four years.

PS. VISTA seems to have sold the rights to the Meca controller to a Mr. Byron Wagner, Genius Co., 1-213-462-1206.

PPS. What about the cassette library? I have been unable to contact Lew Moseley. I now have an 8 inch disk system, since I was counting on Lew for format conversion to VISTA format. If that no longer exists, I would now be able to download to cassette using TAPEDISK, or perhaps in VISTA V-200 format, if no one else is willing or able.

[Editorial remarks:

The Graphic Add was made by K.E.A. Micro Designs in Canada. Not much market for it now. It converts some of the inverse video symbols into 2 pixels x 3 pixels per character graphics. We're putting the manual into *Encyclopedia Processor Technica* Volume 2.

About the piggy-backs versus a new Sol-PC, Bob has mentioned long range plans to make a new PC board, but it will probably be a 16-bit CPU. See my thoughts about Sol upgrades in this issue. The piggy-backs let you add some reasonable new features to Sol, selecting just those you want. Proteus does not own any Sol hardware rights. Has anyone heard from Lew Moseley? —]

...ON USING COMPUPRO 8085/8088 CPU BOARD TO UPGRADE SOL

Dear Stan, 10/16/83

Your articles on the IEEE 696 standard in PROTEUS prompted me to try my CompuPro 8085/88 CPU board in the SOL. It works fine! Of course there is a modification required to the SOL, but NO permanent changes are required. The modification involves removing seven ICs, lifting several pins on two more ICs, and adding three jumpers. These changes can be made without removing the SOL motherboard. In fact - excluding the time to make the jumpers - the SOL can be mod'ed or demod'ed in about fifteen minutes.

Based on your article, I realized that the SOL is close enough to being IEEE 696 compatible that any IEEE 696 CPU board should work in the SOL's backplane if the SOL's 8080 CPU is disabled. And that is all my modification accomplishes. I remove the 8080 and its associated bus drivers, jumper PDBIN to U48 to disable the data line drivers except for data inputs from the SOL's internal bus, jumper the keyboard reset to the S-100 reset, and add one more jumper for housekeeping. That is all it takes!

The 8085/88 board has a power-on jump option which can make the 8085 jump to C000 on power-on or reset so the SOL's four phase wonder isn't needed. Another option allows selection of a 2 MHz clock for the 8085 so the old PT dynamic memory boards can be used - at least for check-out when you are just trying to make the 8085 work. Since I am currently running with a PT 64K board, I have only checked out the operation of the 8085. The 8088 uses a 5 MHz clock. Here there are two possibilities: change the 8088 clock to 2 MHz by changing a crystal on the 8085/88 board or get faster memory and access the display/system memory only with the 8085. I'll probably try the crystal change first and go for the faster memory latter. Since I am using the HELIOS disk drives, I have also shown that DMA without bus arbitration works ok.

I think the CompuPro 8085/88 CPU board is the next logical upgrade to the SOL. As Godbout says, "The CPU 8085/88 Dual Processor board was specifically designed to make it easy for the S-100 Bus user to get into the world of 16 bit micros, while at the same time preserving compatibility with existing hardware and software...

"The user may switch back and forth between the two processors with a simple software command. For example, this allows the user to let the 8085 run his currently available (and familiar) disk operating system while letting the 8088 run the more advanced applications software..."

To modify the SOL: remove ICs U50, U63, U67, U68, U94, U105 and U107; lift pins 3, 5, 7, and 9 on U81; lift pins 3, 5, 9, 11 and 13 on U77; add a jumper from pin 5 of U50 socket to pin 10 of U48 (pin 10 of U48 must be lifted from the socket and the jumper attached to the lifted pin. The easiest way to do this is to solder the jumper to an Augat pin and plug pin 10 of U48 into the Augat pin.); add a jumper from the Augat pin which is RST jack N to pin 14 of U63 socket (RST jack N is just to the right and down from U63); add a jumper from pin 17 to pin 23 of U105 socket. I soldered pins from a dead IC to the jumpers for all jumper ends which need to be plugged into sockets.

To setup the 8085/88 CPU board: DIP switch 1, 1-5 off, 6-8 on; DIP switch 2, 1-6 off, 7 & 8 on; DIP switch 3, 1-5 off, 6-8 on. Dip switch 3 sets the port address for memory management and processor switching. The setting for this is arbitrary as long as it does not match any other port address in the system.

I think this modification is a good start toward making the SOL completely IEEE 696 compatible while bringing the SOL into the 16 bit world. Additionally, this change opens several sockets on the SOL motherboard to be used for extended I/O and memory addressing. One problem with this change is that the SOL's pullup resistors are left on the S-100 bus. Those resistors in parallel with comparable ones on the 8085/88 board make the pullup resistors 600 ohm rather than the specification value of 1000 ohm. This could be fixed by cutting the SOL motherboard traces. However, that commits me to the change. If possible I'd like to retain the option of returning to the original configuration. This can be done by putting Augat pins in the trace on either side of the cut to permit reconfiguration by simply adding jumpers between the pins. I'd like to see all enhancements to the SOL implemented in a similar way so that reconfiguration will always remain viable. As time permits, I'll continue to work on the other changes required to bring the 8085/88 SOL up to full IEEE 696 compatibility.

Over the years I've gotten far more from PROTEUS than I have contributed. Each time I write a letter to PROTEUS I again understand the great effort that you have put into PROTEUS. Your yeoman service to PROTEUS is greatly appreciated by me.

Sincerely,

Ed Bolton

4253 Moore St.
Los Angeles, CA
90066
213-306-7788

Thanks Ed! This letter was a big help to all of us, too.
Stan

Jan 12 1983

...ON A SOL KEYPAD MODIFICATION

PROTEUS
Suite 218
1690 Woodside Rd
Redwood City CA
94061-3483 USA

Dear Stan:

I have expanded my clear memory space to 60K using Fr McGahee's super-phantom and it works well, at last. I'm not much on hardware and I had tried Jim Spann's mod to no avail. Now I just have to get my CONDUCTOR disk controller out of the way from F000 and I can run Whitesmith's C.

How about some articles on IEEE bank switching for the SOL? Surely there are enough loose ports around on the SOL to allow us to page big ram boards around! Speaking of ram I've only gotten two ram boards to fully work on the SOL: the ProTech 64K and the Measurement Systems and Control DM6400. Others I can only get working in the lower 48 (eg Extensys RM64-64). Could we get a compilation of compatible boards and/or necessary modifications? Has anyone modified the ProTech 64K board for the faster operation that the SOL allows? Where can I get the Delay line (to replace the Valor DL 1992, 375ns delay line) or how can I whip one up? Maybe just a 256K mod for the ProTech 64K?

Have their been any BIOS's published which take better advantage of the SOLOS? My BIOS thinks the SOL is a teletype! It would be really neat to have full screen interaction with CPM - eg reexecute a command by simply moving to the end of it and issuing a CR.

Like everyone I'm anxious that my SOL is falling out of date. Any experience with getting a 68000 slave board working? I know I can lift the backplane out and use the SOL as a terminal but without at least a memory board it makes a very poor terminal. There must be some way of interfacing big memory, high speed and sophisticated processors without giving up SOL bus communication.

A FREE, EFFICIENT SOL KEYPAD MODIFICATION

I think I've seen an ad for a keyboard modification which allows the arithmetic key pad to perform some useful function for those of us who do not do a heck of a lot of decimal data keying. In any case here is a mod which is free and relatively easy. It changes the key pad and the Load key to have a hex F as their first nibble:

OLD

```
      < - > < * > < / >
<LDAD> < 7 > < 8 > < 9 >
       < 4 > < 5 > < 6 >
       < 1 > < 2 > < 3 >
       < 0 > < . > < + >
```

NEW

```
      < FD >< FE >< FF >
< FC > < F7 >< F8 >< F9 >
       < F4 >< F5 >< F6 >
       < F1 >< F2 >< F3 >
       < F0 >< FA >< FB >
```

ALTERNATE

An even easier modification allows you to distinguish the keys. But as a hex pad will be more difficult for the software:

```
      < FD >< FA >< FF >
< 8C > < F7 >< F8 >< F9 >
       < F4 >< F5 >< F6 >
       < F1 >< F2 >< F3 >
       < F0 >< FE >< FB >
```

All modifications except two occur on the underside of the keyboard PC except two cuts one of which is easy. The other cut is underneath the keys and would be quite tedious if you actually had to remove all those screws - which I did the first time. The trace can be cut however by simply drilling through the PC board where the trace is. This cut is optional and just changes the LOAD key to part of the Hex pad.

Step one joins the LOAD key logically to the key pad by switching the most significant nibble to F.

1a) Drill through the PC slightly up and to the right of the lower half of keypad 62 (MODE) where you can faintly see a trace on the other side. Be sure to check that the cut is made either by reinserting the keyboard and seeing that MODE, CLEAR and HOME are disabled, or by ohmmeter.

b Reconnect those keys to where they were connected before by jumpering the lower pad of key 70 to the fourth resistor of RX3 (as seen from the underside of the board).

c Cut the incomming trace to the same fourth resistor.

d ==> Essential to either option: cut the trace to the second of those resistors

e Jumper the other ends of the two cut traces to resistors 2 and 4. This ties the LOAD key into the "F" upper nibble with the other pad keys (when step 3 below is completed).

The following optional steps interchange the lower hex nibble of "." and "*":

2a) Cut the trace from the upper pad of key 84 to upper 69.

b Cut the trace leaving the upper pad of key 72.

c Jumper upper pad of 84 to the other end of that trace

d Jumper upper pad of 72 to upper pad of 68

The final step joins the upper nibble of the black and gray pad keys and gives them the F upper nibble.

3a) Jumper the lower pad of 82 to lower 85.

b Jumper the lower pad of 71 to the fourth resistor of RX1 (as seen from underneath). This jumber need only be 3/4" long if you trace its lead back to a hole near 71.

c On the top of the board cut the trace from that resistor to U17 pin 5.

d Jumper that resistor (the fourth) to the unused pin 1 of u17. Do this underneath the board.

Be sure the trace to the second resisitor of RX3 is cut, as mentioned in step 1.

Just by way of application I have included the key assignments for my favorite editor PMATE v3.0. Note that MS is the MODE SELECT key. P<something> is the pad key with the corresponding something written on it. The characters v^<> mean up, down left and right both as actual motion and for the four SOL cursor keys. The little "c" indicates a control character.

```
128 HOME  top/end buffer      148 LOAD  pop garbage
129 CLEAR end buffer          149 P.    command mode
130 cS    <                   150 P0    overtype mode
131 P1    < word              151 cJ    insert line
132 cD    >                   152 <     < geometric
133 P3    > word              153 >     > geometric
134 cW    ^ 1 line            154 P8    ^ mixed
135 cQ    ^ 13 lines          155 P2    v mixed
136 cX    v 1 line            156 cR    move block
137 cZ    v 13 lines          157 LOAD  load block
138 P5    delete char         158 ^     ^ geometric
139 P/    kill line           159 v     v geometric
140 P*    insert mode         160       top buffer
141 MD,MD edit mode           161 P4    < mixed
142 cC    abort               162 P6    > mixed
143 c_    shift case          163 c^    start file
144 cP    redraw screen       164 c]    end file
145 cT    tag position        165       start/end file
146 P-    delete word <       166 c\    toggle case
147 P+    delete word >       167 c0    transpose

PROGRAMMED COMMANDS

P9                    1-m    (end of line)
MD, <1,2,3...0>       .1, .*, ... .<blank>
MD, <P0, P1, ...>     .0, .1, .2, ....
```

Yours truly

Greg Heil, PhD

ANVIL Applied Science
408 Euclid
Toronto Ontario
Canada M6G 2S9

...ON SAM76 OCTOBER 8, 1983

EDITOR, PROTEUS MAGAZINE
1690 WOODSIDE ROAD, SUITE 219
REDWOOD CITY, CA 94061

BERNARD PLOTKIN
5245 N. BAY RD.
MIAMI BEACH, FL
33140

DEAR STAN:

THANK YOU FOR ALL THE BACK ISSUES, WHICH I AM STUDYING AND WRITING LETTERS, WHEREVER I THINK I CAN HELP. ENCLOSED ARE COPIES OF SEVERAL YOU MAY WISH TO PUBLISH.

IN YOUR MAY/JUNE 1980 ISSUE IS A QUERY BY DANIEL HENDRICKS OF IRVINE, CA AND YOU ABOUT SAM76.

VOLUME 34 OF THE CPMUG IS DEVOTED TO SAM76, AND HAS VERSION 23, WHICH I STUDIED FOR ABOUT A WEEK. IT ALSO HAS PROGRAMS AND INSTRUCTIVE MATERIALS. CREATIVE COMPUTING, MAY/JUNE OF 1978 ALSO HAS AN ARTICLE STARTING ON PAGE 30. DR. DOBBS ALSO HAS MATERIAL ON SAM76 IN VOL 3, ISSUE 3, PAGE 46; VOL 3, ISSUE 10, PAE 19; AND THE MAIN DOCUMENTATION IN VOL 3, ISSUE 1, ON PAGE 18. ALSO AVAILABLE FOR $20 IS A WELL WRITTEN HANDBOOK FROM SAM76 INC. BOX 257 RRI PENNINGTON, N.J. 08534.

I WAS ABLE TO UNDERSTAND THE LANGUAGE PRETTY WELL, AND ABOUT 95% OF THE FUNCTIONS I TRIED, WORKED. HOWEVER I WAS NOT ABLE TO TOGGLE THE PRINTER, ALTHOUGH THE BOOK SAYS THIS IS POSSIBLE BY CHANGING A CHANNEL TO LST. IT IS POSSIBLE THAT THE VERSION I HAVE FROM CPMUG VOL 34 IS NOT THE LATEST.

...ON THE SOL PENCIL PROGRAM *OCTOBER 9, 1983*

BOB JOHNSON
565 MOHAWK DR. A-5
BOULDER, COLORADO 80303

BERNARD PLOTKIN
5245 N BAY ROAD
MIAMI BEACH, FL
33140

DEAR BOB:

I SEE YOUR MAY 6 1981 LETTER TO PROTEUS ASKING ABOUT HELP IN GETTING YOUR VERSION OF PENCIL TO PASS CONTROL CHARACTERS TO YOUR EPSON MX-80.

I HAVE A SOL PENCIL CASSETTE PROGRAM, WHICH I CHANGED TO USE WITH MY CP/M DISKS. I HAVE DISASSEMBLED MY VERSION AND UNDERSTAND IT PRETTY WELL. ALL 26 CONTROL - ALPHABET CHARACTERS ARE USING BY IT FOR PENCIL FUNCTIONS. THUS ANY PRINTER CONTROLS YOU WISH PENCIL TO SEND ARE INTERPRETED AS PENCIL FUNCTION SPECIFICATIONS. FURTHER PENCIL PLACES YOUR TEXT IN A BUFFER, AND MANIPULATES THIS AS NEEDED TO PROPERLY APPEAR ON YOUR VIDEO. I BELIEVE THE LOGIC DIVIDING UP THE SENTENCES COULD NOT MAINTAIN CONTROLS, AS EVERYTHING IS ASCII. THE PROOF IS THAT WHEN IN PENCIL YOU ARE NOT ABLE TO SEND A CONTROL CHAR TO THE SCREEN. IF WE CAN DO THIS, IT MEANS IT IS IN THE BUFFER. I AM THINKING ALOUD. AND SINCE I USE WORDSTAR, WHICH IS EASY TO INTERFACE WITH THE PRINTER CONTROLS, I DON'T MISS IT. THE ONLY THING I CAN THINK OF IS TO SACRIFICE ONE PENCIL FUNCTION AND USE IT TO SET THE PRINTER. THE OTHER THING WOULD BE TO TAKE SOME OF THE ODDBALL CHARS LIKE ~,{,}, AND USE THEM AS PRINTER CONTROLS. THEN PLACE A FILTER ON YOUR PRINT OUT ROUTINE, AND INTERCEPT AND REPLACE WITH THE PRINTER FUNCTIONS.

HOPE THESE MUSINGS HELP. I AM RUNNING TWO THINKERTOY DISKS WITH 48K RAM, AND LOTS OF PROGRAMS. IF YOU HAVE CPM, WOULD BE HAPPY TO SHARE WITH YOU.

...ON DEAD KEYS OCTOBER 8, 1983

EMILE ROTH
1001 EVELYN TERRACE EAST, #104
SUNNYVALE, CA. 94086

BERNARD PLOTK.
5245 N. BAY RD
MIAMI BEACH, FL
33140

DEAR EMILE:

I NOTE IN A BACK ISSUE OF PROTEUS THAT YOU WERE TROUBLED WITH DEAD KEYS ON YOUR SOL. PROBABLY YOU HAVE ALREADY FIXED THE TROUBLE. IF NOT, MY EXPERIENCE SHOWS IT IS POSSIBLE TO REPLACE THE FOAMPAD.

THE AIR-CONDITIONERS IN MY HOME COME WITH URETHANE FOAM BLOCKS AS INSULATION AND AIR STOPS. I TOOK A SMALL PIECE AND CUT OUT A CYLINDER ABOUT 7/16 INCHES IN DIAMETER AND 1/2 INCH LONG. I THEN CUT OUT A 7/16 INCHES CIRCLE OF HEAVY ALUMINUM FOIL. I USED THIS TO REPLACE THE ORGINAL FOAM, USING SIMPLE WHITE GLUE.

THE WORK IS DONE FROM THE BACKSIDE, AND REQUIRES MOVING ABOUT 16 SCREWS, TO REMOVE THE CIRCUIT BOARD. THEN THE KEY CAN BE TAKEN OUT BY PRESSING ON PLASTIC TABS THAT HOLD IT IN THE SLOT.

IF YOU CAN GET A SMALL COPPER TUBE WITH 7/16 INCH INNER DIAMETER, I THINK THAT WOULD BE HELPFUL. I COULDN'T, AND MY DIMENSIONS WERE NOT THAT EXACT. STILL IT WORKS.

...ON NETWORKING

James D. McElroy
2026 Crest Ave. N.
Allentown, PA 18104

October 23, 1983

Stan, I understand your feelings about the evolution of the SOL and of Proteus. I agree that the advent of powerful single board computers and enormous memory capacity tend to make the SOL look feeble by comparison. However, the reasons that I bought the SOL to begin with are still valid. The keyboard is still one of the best, and I can read the display with my trifocals! And the walnut sides have class!

I think that the networking of computing power is one of the important developments of the evolution of the personal computer. Your suggestions amount to having a local network in our familar blue boxes. With you on the West Coast, and Fr. Thom Magahee on the East Coast, we all have a lot to look forward to.

Dear Proteus, ...ON DISK CONTROLLERS AVAILABLE

I wish to add a disk drive and a programmable character generator to my 48K Sol system. From your extensive resource pool, what are the most popular/best/easily interfaced devices available. Micropolis seems to be able to be tailored for Sol most easily. Objective Designs PCG is the only one I've come across that works with Sol specifically. Please comment.

C. Jason Slade
32 Berkley Cres.
Simcoe, Ontario N3Y 2K4
Canada

[Editor's reply: For 5-1/4" disk, I recommend the MicroComplex disk controller which Proteus sells as the "McFloppy" controller. It is NorthStar compatible, but is a superior hardware design since it is all digital for reliability and handles up to 96 tpi drives for super capacity. See the description in the Proteus catalog in last year's Proteus News, or write us for a copy. Bob Hogg customizes the software for Sol. Micropolis is popular among East coast Sol owners.

I'll be surprised if you can find a new Objective Designs PCG. Maybe our readers are familiar with others.]

20

TABLE OF CONTENTS

SOL AND THE IEEE 696 STANDARD--PART 2:
UPGRADING THE SOL, COMPLIANCE OF MEMORY AND
PERIPHERALS by Stan Sokolow 1

MEET ADA, THE NEW PROGRAMMING LANGUAGE by Stan Sokolow.....1

CLARIFICATION OF THE DUAL PERSONALITY MODULE by Stan Sokolow..2

INTERESTING ADVERTISING WE'VE RECEIVED by Stan Sokolow 2

SOL'S CO-DESIGNER ON COVER OF INFO WORLD.....3

THIS 'N THAT BOARD by Neil Sutcliffe 9

OFFICE SYSTEMS RESEARCH ASSOCIATION CONFERENCE 14

SAY, WHAT'S BOB MARCH UP TO THESE DAYS?.....15

PTC BASIC RISES AGAIN AS "NEVADA BASIC".....15

ELLIS COMPUTING, INC., Press Releases.....15

LETTERS TO THE EDITOR:
...ON A DATA BASE MANAGER AND ELECTRONIC SPREAD SHEET
by Roy Heybrock.....15
...ON AMATEUR RADIO SOFTWARE FOR SOL by Francis Biasiotto...15
...ON PT BOARD DOCUMENTATION by Ed Rainsberger 16
...ON DISC DRIVES OTHER THAN HELIOS by Harold Lyons 16
...ON QUESTIONS ABOUT DISK CONTROLLERS by Jason Slade 16
...ON A REAL TIME CLOCK INTERFACE by Bruce Bull 16
...ON ECB FILES WITH G/2 BASIC by John Barber 17
...ON USING COMPUPRO 8085/8088 CPU BOARD TO UPGRADE SOL
by Ed Bolton.....17
...ON A SOL KEYPAD MODIFICATION by Greg Heil 18
...ON SAM76 by Bernard Plotkin 19
...ON THE SOL PENCIL PROGRAM by Bernard Plotkin 19
...ON DEAD KEYS by Bernard Plotkin 19
...ON NETWORKING by James McElroy 19
...ON DISK CONTROLLERS AVAILABLE by Jason Slade 19

NOTICE TO MEMBERS

VISA and Master Charge for Proteus Orders will no longer be available after 1983. Newsletter format may also change to 8 1/2 X 11 xerox copy. These changes are due to financial and staffing difficulties. However, be assured that Proteus will remain a source of materials for and a vital communication link between all interested SOL owners.

Z-80 CONVERSION "McZOL" IS IN PRODUCTION

Proteus has received the first Z-80 module for upgrading the Sol without using any slots. Bob Hogg once again has made a superb engineering design for retrofitting the Sol-PC by piggy-backing. It works in conjunction with the 24x80 upgrade (McVideo) and dual personality module (McDPM).

I'll be testing it in my Sol and will write an article on it for the next newsletter. Due to the lateness of this issue, the next issue of Proteus News will be slimmer to expedite getting the news out on the Z-80 module.

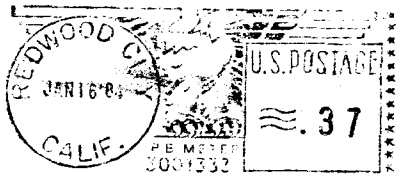
PROTEUS/NEWS

A news journal for owners and users of Processor Technology Corporation computer equipment. Published by Proteus, 1690 Woodside Road, Suite 219, Redwood City, California 94061-3483,USA, telephone (415) 368-2300.

Submit items for publication to Proteus News, Attn: Stan Sokolow, 1690 Woodside Road, Suite 219, Redwood City, California 94061-3483, USA. Please make submissions as camera-ready as possible by using a fresh, black ribbon and typing single-spaced.

FROM:
PROTEUS
1690 WOODSIDE ROAD, SUITE 219
REDWOOD CITY, CALIFORNIA 94061-3483
USA

FIRST CLASS MAIL



James D. McElroy
2826 Crest Ave. North
Allentown, PA

18104

PROTEUS / NEWS

AN INDEPENDENT NEWSLETTER FOR OWNERS AND USERS OF PROCESSOR TECHNOLOGY CORPORATION COMPUTERS

FORMERLY SOLUS NEWS

1983
Vol. 6 #3

PUBLISHED BIMONTHLY BY PROTEUS, 1690 WOODSIDE ROAD, SUITE 219, REDWOOD CITY, CA 94061, USA

SINGLE ISSUE...
$7.50 (US)
$9.50 (FOREIGN)

EDITOR'S MESSAGE

This is the new look for Proteus News. To help reduce the production cost as our readership slowly declines, I've changed to a direct printing method that doesn't require the more costly photo-reduction. This will simplify production of the newsletter.

CRYSTAL BALL GAZING

UNIX IN THE NEWS

It's always fun to try to guess what's happenening to microcomputing in the future. Here are some of the recent visions that came in on my crystal ball.

Digital Research Incorporated (DRI), the system software company that brought you the CP/M family of operating systems, has announced that it is transporting (otherwise known as "porting") the UNIX System V operating system onto the Intel 286 microprocessor under an agreement between Intel and DRI.

UNIX is the popular operating system developed for internal use by AT&T employees at the Bell Labs facility, but it became the joy of many college students because AT&T was allowed to distribute it to universities. Since AT&T is deregulated, the new AT&T Technologies company that grew out of Western Electric is gearing up for competition with IBM in the computer market. Naturally, it is to AT&T's advantage to promote UNIX as a standard operating system for the 16 and 32-bit microprocessors.

A DRI newsletter for Independent Software Vendors (ISV Forum Vol 4, No 1) headlines the idea that UNIX System V is to be promoted as a standard. DRI and AT&T have agreed to develop a UNIX System V Applications Library to be marketed through DRI's retail outlets.

DRI has already released versions of its CP/M compilers that run on UNIX. This allows ISV's to recompile their applications on a UNIX machine and create an instant new market. DRI is counting on the thousands of applications it fostered in the CP/M world to help it build a UNIX library quickly.

What about IBM in all of this. It is well known that IBM owns a substantial chunk of Intel and has a director seat on the Intel board. IBM chose Intel as a rapid way to pick up microprocessor technology for its PC products. It is rumored that IBM is developing a small business multi-user computer based upon the Intel 286 microprocessor. Will it use UNIX?

Based upon prior history with the IBM-PC and the new wrinkle of AT&T competition, my crystal ball has been receiving visions that say the IBM multi-user microcomputer will offer UNIX as an extra-cost option, but a unique operating system developed by or for IBM will be the standard OS with the machine. It may even be an enhanced version of the MicroSoft IBM PC-DOS.

I can't see IBM giving support to AT&T by dependence upon the UNIX operating system. IBM will want to avoid paying large royalties and being at the mercy of a competitor.

Meanwhile, DRI will try to cover all bases by continuing to extend its family of operating systems to other processors and to provide new features, such as windowing. It is also porting its compilers to competitors OS's, specifically MS-DOS and PC-DOS.

But, my guess is that DRI really is hoping to be the prime vendor for the AT&T world, since it lost the battle for IBM to MicroSoft.

ALIGN YOUR OWN DISK DRIVES

by Stan Sokolow

I own six Sol/Helios systems, all but one of which I picked up at quite reasonable prices after PTC went out of business. These have served me faithfully for word-processing and accounts receivable mostly, with perhaps one breakdown a year among the six.

But the fellow who has been doing my disk repairs has gotten progressively busier himself, and the turnaround ground to a halt. Over a several month period, five of the systems went out of service, leaving me on the brink of disaster. (Well, not disaster, because I have carefully planned a fail-safe manual system to fall back upon in the case that none of the machines worked. But, it sure would be inconvenient.)

I tried a few other disk repair places, but most of them were not able to repair the PerSci drives used in my Helios systems. I found a place that had the equipment and know-how (they said), but when I sent my drive, they let it sit on the shelf for several weeks and finally returned it unrepaired -- they said their PerSci exerciser was broken.

I discovered that the remains of PerSci, Inc. is a skeletal crew of the former manufacturer. They will service drives, but their minimum rate is $400 per drive! With five drives out, that was a minimum of $2000.

Well, this was enough motivation for me to get off of my duff and see if I could do the drive repairs myself. I have the PTC technical manual (Encyclopedia Processor Technica volume 9) which describes the alignment procedure and gives the schematics and some repair guidelines. I rented an appropriate oscilloscope and sat down to try it.

I'm pleased to report that even mere mortals can learn how to align a PerSci drive! I have worked on them in spare time on two weekends and have 3 of the 5 back in service. The 4th has a problem which I've zeroed in on and expect to fix it this weekend. The 5th is the hardest because it needs new bearings.

I want to give you the results of my experience.

First, let me shatter a myth. Not one of the five drives was "out of alignment". I mean that I always thought the term "alignment" meant whether the head was properly lined up in the standard location on the diskette. In the jargon of the disk drive world, this is called radial alignment and azimuth. None of the drives had this problem.

What was wrong then? A variety of little things broke down. One drive wouldn't seek -- its positioner lamp had burned out. One got persistent ABORT errors when testing with DISKT test program -- it had a bad TTL chip in the data separator board within the drive. Another also gets persistent ABORT errors -- it has a bad TTL chip on the data & interface board in the drive. Another has a bad power supply in the Helios box (the +5 volt regulated supply is giving only +3.2 volts). And the fifth one has the bad spindle bearings and the positioner mechanism whistles (oscillates).

Now, the positioner oscillation is often due to a weak positioner lamp, and this can be adjusted by twisting a screw on a potentiometer in the drive (unless the lamp has gone too far). But other than that, all of the other problems were just worn out electronic components, not "alignments". They simply needed repair.

In doing the repairs, though, it is necessary to check and readjust all of the alignment steps. In fact, that's how I discovered the parts that were bad. I couldn't adjust the signal to meet the specification, or there was no signal at all.

In the greater sense, alignment also means adjustment of the electronics to meet operating specifications, so in this sense many drive problems may be solved by "alignment". As components age, their characteristics change a little, and adjustment of the circuit may bring the drive back into working condition.

However, eventually something is going to break down. So, think of repair rather than just alignment or adjustment. If you have a disk drive made by a company still in operation, you're in luck. If your drive is an orphan, you may be faced with the choice of paying through the nose, junking the drive, or treating it as a learning experience as I did.

I also discovered another bonus in doing it myself. I suspect that most repair facilities simply get the drive back into operation and check only the most common alignment specifications. On my drives, I found several with the index photo-detector out of specification. This did not affect their passing of all the other tests.

I have had a nagging problem with the drives over the past several years which has only happened once a year, but which was devastating when it hit. I now suspect that the index timing was the culprit in this bug (which resulted in the drive wiping out the first header on a track when it wrote a long sector at the end of the track). So, if you want to be sure it is done thorougly, I guess you have to do it yourself.

What do you need to be able to do alignments of the PerSci drives? (I suppose alignment of the other brands of drives would be similar but easier because they use simple stepper motor positioning rather than the faster, more complicated, voice-coil positioner of the PerSci.) You need a dual trace oscilloscope with at least 25 MHz bandwidth, external trigger, and the ability to display the difference between the two channels (add with inverting channel B). You need a Dysan 240A alignment diskette. You need a few small hand tools like screwdrivers, plastic bladed screwdriver, allen wrenches, etc. You need the SIMU-ciser program written by Processor Tech, or a similar program you write that will step the drive to whatever track you want or automatically step at full speed between two designated tracks. And you need the manual I mentioned.

You also need to understand how to read the schematics if you are going to have any hope of doing repairs, rather than just adjustments. I have no engineering background, but I've picked up enough electronics understanding over the years since I bought my computer by reading things like the Don Lancaster TTL Cookbook, the Radio Shack "Understanding Solid-State Electronics" book written by Texas Instruments, Byte magazine, the Sol manual, Proteus News, etc.

I rented my scope from Continental Resources, which has home office at 175 Middlesex Trunpike, Bedford, MA 01730 (telephone 617-275-0850). They have a branch in Santa Clara, which is where I got mine. You probably won't find a scope with exactly those specifications, so treat them as minimum requirements. Look for a rental company under "Electronic Instruments -- Rental" in the yellow pages directory.

The cheapest scope that was on hand when I called to rent was a Tektronics 2215, which is 60 MHz dual trace and has other fancy features, but it does the job. It rents for $100 per month, but I've rented a similar but simpler scope for $80 before. You should ask to rent a third probe, which you'll need for the trigger signal.

Now, some of you may have heard about the Dysan Digital Diagnostic Diskette (DDD). This is a device that lets you check alignment of the drive without an oscilloscope. There was a good article on it in Dr. Dobb's Journal, December 1983. Why didn't I use it?

As I mentioned, my drive problems turned out not to be "alignment" problems, but rather electronic problems for the most part. The DDD is of no help in trouble-shooting a faulty circuit. Also, the PerSci positioner mechanism is different from that in other drives. To adjust it and check its performance, it is necessary to look at certain voltages at test points on the board. But also, the DDD is designed to be read by a standard soft-sectored disk controller, which the Helios is definitely not. (But the main reason I didn't use the DDD was that I wanted to do it by the book since I was a novice. This turned out to be a wise decision.)

UNCLASSIFIED ADS

FOR SALE:

One 80 column modification board with all chips included, plus SOLOS personality module 80 column ROM chip (Personality Module not included). Board designed and produced by Bob Hogg for SOL computer. Provides 80 column VDM output. New Personality Module for use with the ROM chip may be purchased from Hogg or through Proteus. Original cost $295US. Never used. Sale price $$225.00US. Shipping included.

WANTED:

To buy old SOLs for parts. No memory boards, add-on components or accessories wanted. Must be in working condition, especially keyboards and backplanes. Reply with price, inlcuding packaging and delivery.

Please call or write to: KUZIAK CONSULTING, 100 Connaught Crescent, Regina, Saskatchewan, S4T 6M9, Canada, 306-527-0154. 9 a.m. to 12 a.m., 1 p.m. to 5

FOR SALE

SOL 20, (Rev.E), with manuals and original box-$325+shipping; Central Data Corp. 32K board (expandable to 64K with 4116's, 2117's, etc.) with manual-$60+shipping; GPM board with ALS-8, SIM-1 & TXT-2, all manuals and updates-$35+shipping. All the above for $375+shipping. All equipment above in very good condition, hardly used, and just kept as a backup system for my main SOL systems:

Curt Kobylarz-Schmidt
1710 N.Wisconsin Ave. Peoria, IL 61603 (309)682-5258

ON A DAISY WHEEL PRINTER AND A RAM DISK by Rick Downs

9 February 84

PROTEUS
1690 Woodside Road, Suite 219
Redwood City, CA 94061-3483

Dear Stan,

It's been a while since I've written about my ongoing experiences with my SOL, so I thought this would be a good time to catch up. I have enclosed a check for $30 to cover my 1984 subscription. I thought I renewed for 1983, but only received one issue(#1). Could you check your records, and mail any back issues - thanks.

I have a couple of new products that I would like to report on. The first is a low-priced daisy wheel printer. The one I'm referring to is the DWP-210 from Radio Shack. Although, I've never been a fan of Radio Shack computer equipment, the past few years have seen dramatic improvements in their products. One such product is the DWP-210.

The DWP-210 is a full daisy-wheel printer with all of the niceties you would expect in a printer costing twice as much. Although it is not as fast as the high priced units, the quality of the print is superior. That was one feature that I was looking for when I went shopping for my daisy-wheel. The impression of the characters is far more important then the speed. The unit is solid - weighting in at 38 pounds. This should tell you something about the quality of construction of the 210.

The printer types at 20 cps average, although that figure can change depending on the type of document you are printing. I have found that it is considerably slower when printing a file that is all CAPS. However, for a normal business letter the speed is very satisfactory. The pitch is both hardware and software controllable. The pitches available are 10, 12 an PS(Proportional Spacing). A switch on the top panel of the printer selects between the three. There is also an On-line/Off line switch.

The 210 uses standard Diablo ribbons, both the fabric, and multistrike. Ribbons can also be purchased at any Radio Shack Computer Center, and selected office supply stores. There are two interfaces that come standard with the unit. An RS-232, and the standard Centronics type parallel port. I'm currently using the parallel port - so I had to customize my own cable - which was not that big of a deal.

There are three warning sensors on the printer. The first is a "paper out", the second "replace ribbon", and the third indicates that the top cover is open. When turned on the printer initiates a self-test routine that checks out the printer mechanism, carriage, etc. This all takes about 3 seconds. There are also some DIP

switch selectable options such as BAUD rate(600/1200) for the serial interface, self test, logic seek, serial/parallel interface, etc. The switch is located under a small sliding door on the top of the printer in the back. This allows easy access, though generally it is not used once the initial settings have been selected.

Another real nice feature of the printer is that it is bidirectional. This makes the total throughput somewhat more efficient. Certainly a bonus for a printer in this price range. The only option available at this writing is a bidirectional tractor feed. The printer is priced at $799 and the tractor feed is $150. One irritating feature that is lacking is the means to turn off the CR/LF sequence mechanically. This can done under software control though. I have successfully patched both Wordstar, and Electric Pencil to not issue an extra line feed. I also patched Wordstar to recognize all of the features of the printer - boldface, ~~overstrike~~, underscore, etc.

I am very satisfied with this new addition to my computer peripherals. It has worked beautifully for over 9 months without any problems. Radio Shack products do deserve a second look.

Another product that SOL owners will be interested in is the RAM disk from Digital Research Computers of Garland, Texas. The Light Speed Disk Simulator or LS-100 is a 256k byte RAM board(S-100) that emulates a disk drive. The software uses CP/M as the operating system, although other disk system may well be patched to operate it as well.

The LS-100 board uses the new 4164 64k x 8 bit DRAMS(Dynamic) chips. It also uses a new Dynamic Ram controller chip that is manufactured by Intel, the 8203-1. This eliminates a lot of the timing problems previously associated with dynamic RAM boards. It has been working fine in my highly modified SOL for the past 6 months.

The LS-100 is available one of two ways: (1) the complete kit of all parts, including the S-100 board and software on an 8" IBM format disk, or (2) a bare board with the same software. The complete kit is $399, while the bare board is $69.95. I opted for the bare board, and managed to save about $100 over the kit price. However, there is always a tradeoff in order to save money. If you decide to build the board from scratch the Intel controller will be hard to find. It is also the single most expensive component - expect to pay around $45. I finally found one through my local Wyle Labs distributor.

Once the board is operational, the software must be properly patched to your CP/M system. This was not as difficult as first thought. More specifically, a few EQUATES need to be changed in the supplied software to that of your CP/M size, etc. After the software has been patched and reassembled, following the routines described in the manual the board is ready to go.

There are three basic programs used with the board. The first is a diagnostic(DIAG) program that checks the functionality of the board

and also performs a memory test of the chips. Once executed, the program will display a memory map indicating a good(G), or bad(B) memory chip. This need only be run if you suspect a failing memory chip or get a checksum error during install.

The other two programs must be run before the board can be used. The first program is a format routine(FMAT) that formats the board, just as you would a regular disk. It takes about 3 seconds to format the board. The second program is the install routine(INSTALL) that actually allows CP/M to access the LS-100. When this is run a signon message will appear noting that the LS-100 drive is active. At that time files can be PIPed over and that board accessed as a regular disk drive.

I have used several word processors, spelling checkers, and assemblers with it and the performance is really - unbelievable. To give some timing comparisons I used a 600 unique word Electric Pencil file, and checked the spelling using The Word Plus. With a regular 5 1/4" floppy, the time was 2.5 minutes. With the LS-100 the time was reduced to 19 seconds. That is a dramatic improvement!

I am very pleased with the overall performance the LS-100 has added to my system. It even has battery backup capabilities that would allow you to always be on-line. In fact as many as eight of the these boards can be installed in the same system giving you a total of over 2M bytes of on-line storage. I wander if the new 256K x 8 bit chips can be substituted?

Well, I'd better stop and give someone else room in the newsletter. Thanks for your continued efforts on the newsletter. I'm still traveling quit a bit so I don't spend as much time as I would like at the keyboard. However, I plan to write more about other applications I'm working with that may be of interest to other SOL users.

With all the new computers on the market today, and the proliferation of PC compatible machines, sometimes I feel left out. Everything I know about computers, is because I gambled on PT way back in 1975. I guess we're a dying breed - now that the little MAC is out.

Regards,

Rick

Rick Downs
Denver Colorado
(303) 750-1838 evenings

ON BASIC/Z by David Dalva

Dear Stan,

I have been using a new Basic called Basic/z written by Bob Zale of Systemation. This has to be close to the ultimate Basic in capability, speed of execution, and ease of use. There are 232 key or reserved words in the current version. Which is 1.11

Among one of the more annoying problems that I have had in MBasic in the my cursor keys are not usable in editing, cursor positioning is not as direct or as easy as it was with some dedicated Basics, reverse video and other terminal characteristics require the use of CHR$ characters which are terminal dependant. Basic/z has commands to clear screen, formfeed printed pages. It is able to determine where the cursor or print head is at any time so that you can write code that if the cursor is at a particular location to take a specific action or if a procedure is interrupted to return, after an action, to the original location. You may through the use of UPCASE statement force all entry to be in upper case. Basic/z supports both CHAIN and COMMON. It does not require, as does MBasic, the use of LPRINT with each line to direct output to the printer. It allows a simple statement, LPRINTER, to direct the output to the printer, while CONSOLE sends output to the screen. This is very similar to the SET OF = statement in PTBasic.

Basic/z has a number of very interesting feature, among them is that it is an interactive compiler, that is supports two different modes of execution. First, in the testing mode, you compile and immediately execute the program. When you have the final version you may execute a one step BIND command which will link the the object program with the needed support routine into an executable command (.COM) file. For maximum efficiency and security, all generated code is in the form of native machine instructions.

Another feature is that each line tested for correct syntax as the line is finished. If there is an error the editor points to the position of the syntax error. There are seventeen local edit commands that allow movement any where within the edit line. The edit commands, all of which are invoked by one key, include advance, back-space, tab, abort, append, change, delete, insert, jump, kill, zap, replace, list, quit, search and move(go directly to the error in the new line).

Basic/z has program definable precision in decimal (BDC) math from 6 to 18 digits.

Some of the more significant keywords are DEBUG, EDIT$, FMT, LABEL, LINK, SORT, SPOOL,

DEBUG - To select various debugging options.

```
DEBUG "L", *, *
```

Used in conjunction with the &D option at compile time would specify entire program and would list the line where the first error occurred. Other options are Pause, Trace, Single-step.

EDIT$ - Allows user editing of a string value, may also be used for input and will place a default value in string.

```
text$ = EDIT$(text$,20)
```

Will print old or default value of text$, will accept correction or new entry of up to twenty characters and

will then change value to new entry. During editing you may use cursor keys for position, insert, delete, and for special actions such as help.

FMT - return a formatted string from a numeric value

```
telephone$ = FMT(telephone, tel.mask$)
```

when

```
tel.mask$ = "(999)  999-9999"
telephone = 3124818085
```

Will out put telephone$ = (312) 481-8085

LABEL - To create a symbolic destination

```
LABEL @print.routine
```

Used in conjunction with GOTO, GOSUB, RESTORE and the like. By use of descriptive labels and variable names it almost possible to eliminate use of REM statements. Basic/zwill recognize any length variable name and all characters are significant. The keyword LABEL is not necessary and **@with.any.name** may be used instead.

LINK - To load and transfer control to an object program.

```
LINK  "A:BZ.COM"
```

Will revoke the Basic/z compiler

SORT - To sort an array

```
SORT Text.element$(0),Text.element$(1024),"DU",12
```

Will sort the first 1025 elements of the array Text.element in descending order forcing the characters into upper case for comparison only. Only the first 12 characters will be compared.

SPOOL - To direct printed output to a file

```
SPOOL 19
```

All output from the LPRINTER statement will be sent to file 19.

Some other keywords are:

ECHO - Print on both screen and printer.

INCR - To increment a numeric variable.

INCLUDE - Allows additional source program to be include in the compilation.

LOWCASE$ - return a string in which all upper case is changed to lower case.

SELECT - To select a default drive.

PAGESIZE - To allow program control of printer page size.

Also are supported are WHILE - WEND, DO - UNTIL, IF - THEN - ELSE

INCHAR$ - permits recognition of a single character.

INKEY$ - will allow a specific number of keystrokes. Which will not be echo on the screen.

The only slight deficiency is the lack of detail examples and suggestions on using some of the more complex keywords.

Basic/z is available from:

System/z, inc.
P.O. Box 11
Richton Park, IL 60471

Sincerely;

David 1. Dalva II

ON A HARD DISK AND CP/M by David Dalva

Dear Stan,

I have received Bob Hogg's new Z-80 update for the Sol and have been using it for the last month. It is completely transparent to the user except for two important features. First, you can run Z-80 software, which seems to be replacing 8080 for machine dependant code. Second, the processor can be run at either 5 mhz or 6 mhz instead of 2 mhz. At this speed the Sol is a absolute joy, as the screen updates almost instantaneously, additionally you can change the processor speed under software control. Also there are available, as an option, non-maskable interrupts to enable single stepping of software under development. . There are some necessary mods to be made to the Sol mother board in order to achive this speed. I sent my mother board to Bob so that he could do both the modifications and testing as some of the chips are critical at the higher speeds.

It is also possible to modify the controller boards for the Helios to work at the higher speeds. At 5 mhz the Helios is very fast. So far, I have not experienced any difficulty in using my old Helios software. Also I have also obtained from Bob a new keyboard enclosure, with a extension cable, which allows me to place the Sol itself out of the way and just have the monitor and keyboard on my desk.

Bob has also supplied me with a 15 meg hard disk which I use in conjunction with his Northstar compatible 5.25 inch drives. After seven months operation I am convinced that this is the only way to go. There is a noticeable speed up on disk access and disk-intensive compilation are noticeable quicker. I have just about completely switched over to CP/M and use it for almost all my work. The big problem is that many of my original programs were witten in PT Disk Business Basic which allowed both CHAIN and COMMON.

Now, there is a PTBasic for CP/M is available from Ellis Computing. At this time they have only implemented the standard PT Disk Basic but have promises to add the CHAIN and COMMON features of the Business Basic. To convert programs you have to save them in ASCII, move them to CP/M and then run under Nevada Basic which is Ellis name for PT Basic. Ellis has also included a full screen editor which is invoked when you generate an error. Nevada Basic also uses the same abbreviation that PT Basic used. I have converted a few programs and have found no significant problems, but I am waiting for the new version, as most of my programs were written in Business Basic to take advantage of its CHAIN feature. Nevada Basic is available from Ellis Computing, 3917 Noriega Street, San Francisco, CA 97122, (415) 753-0186. The price is $39.95. They have also converted PT Fortran, PT Pilot and PT Edit to run under CP/M. THe Fortran is the same implementation that was used by PT and is written by the same author. I have used both Nevada Fortran and Microsoft Fortran and found that there was no sufficient advantage to the Microsoft over Nevada.

I have modified WordStar to use the various function keys on the Sol keyboard.

The modifications are implementated as follows:

In version 3.3 of Wordstar, the Installation Menu offeres you the following choices 'A,B,C,D,E,F or X'. If you enter a '+' instead you will enter the Custom Modification routine. You may make changes by either entering the Wordstar symbol proceeding by ":" for example ":HITE"

In earlier version you can use DDT to effect the changes. The changes are as follows. They presume an 80 by 24 screen and Bob Hogg's operating system and the Solus at F000.

Symbol name	Version 3.00	location 3.3	Code old	in hex new	
HITE	0248	0232	18	18	24 line height
WID	0249	0233	50	50	80 character width
CLEAD1	024A	0234	02	02	number of characters
	024B	0235	1B	1B	ESC
	024C	0236	3D	3D	"="
LINOFF	025E	0248	20	20	offset for line
COLOFF	025F	0249	20	20	offset for column
ASCUR	0260	024A	00	00	select binary
ERAEOL	026D	0250	00	1F	clear to end of line
IVON	0248	0267	00	01	number of characters
	0249	0268	00	0F	turn on highlight
IVOFF	028B	026E	00	01	number of characters
	028C	026F	00	14	turn off highlight
USELST	02AA	0280	00	FF	enables last column
MEMAPV	02E0	0290	00	FF	enable memory-map
MEMADR	02B1	0291	00	00	lo-bit
	02B2	0292	00	F8	hi-bit
HIBIV	02B3	0293	00	FF	set for highlight
HIBCUR	02B4	0294	00	FF	set for cursor
CRBLIV	02B5	0295	00	FF	blink cur in inverse
DEL3	02D1	02B1	19	09	shorten blink
DEL4	02D2	02B2	40	05	shorten delay
DEL5	02D3	02B3	09	00	shorten delay
ITHELP	0360	034D	03	02	suppress main menu
ITITOG	0362	034F	FF	00	insert off at start
NONDOC	0392	0378	00	FF	set to non-doc mode

NE: To suppress SIGNON and Warning in Wordstar 3.3, enter 00 at 02E2h, 3F1Dh and 411Fh.

T A B L E O F C O N T E N T S

EDITOR'S MESSAGE by Stan Sokolow 1
CRYSTAL BALL GAZING: UNIX IN THE NEWS by Stan Sokolow 1
ALIGN YOUR OWN DISK DRIVES by Stan Sokolow 1
UNCLASSIFIED ADS 3
ON A DAISY WHEEL PRINTER AND A RAM DISK by Rick Downs 4
ON BASIC/Z by David Dalva 7
ON A HARD DISC AND CP/M by David Dalva 9
QUESTIONS FROM MEMBERS TO MEMBERS 12

B A C K P L A N E R E F U N D S

Bob Marsh has been too busy to have the backplanes manufactured. We don't want to hold your funds any longer. We will send refunds soon. Sorry.

P R O T E U S / N E W S

A news journal for owners and users of Processor Technology Corporation computer equipment. Published by Proteus, 1690 Woodside Road, Suite 219, Redwood City, California 94061-3483,USA, telephone (415) 368-2300.

Submit items for publication to Proteus News, Attn: Stan Sokolow, 1690 Woodside Road, Suite 219, Redwood City, California 94061-3483, USA. Please make submissions as camera-ready as possible by using a fresh, black ribbon and typing single-spaced.

FROM:
PROTEUS
1690 WOODSIDE ROAD, SUITE 219
REDWOOD CITY, CALIFORNIA 94061-3483
USA

FIRST CLASS MAIL

PROTEUS QUESTIONNAIRE #1

To determine the direction our members would like Proteus to take in the coming year, please complete the following questions and return to Proteus, Attn: Stan Sokolow, 1690 Woodside Road, Suite 219, Redwood City, CA 94061. You may remain anonymous or sign the questionnaire. The tabulations will be published in Proteus News. Thanks for your active participation.

1. Describe your system:

( ) Sol	( ) CUTS in another computer
( ) Northstar floppy	( ) Micropolis MacroFloppy
( ) Micropolis MetaFloppy	( ) Helios floppy
( ) Standard 8" floppy such as Tarbell, Discus, etc.	
( ) Other storage medium:	
( ) Daisy-wheel printer	( ) Selectric
( ) Dot-matrix printer	( ) Other printer

Number of kilobytes of RAM:

2. Software you own:

( ) Extended Cassette BASIC	( ) Microsoft BASIC
( ) ALS-8	( ) CP/M for standard 8" disk
( ) CP/M for Northstar	( ) CP/M for Micropolis
( ) CP/M for Helios	( ) Northstar DOS and BASIC
( ) Micropolis MDOS and BASIC	( ) PTDOS for Helios
( ) Microsoft FORTRAN	( ) Proc. Tech. FORTRAN
( ) Proc. Tech. PILOT	( ) Other PILOT
( ) UCSD Pascal	( ) Other Pascal
( ) COBOL	( ) Electric Pencil
( ) WordWizard	( ) TEX CP/M text processor
( ) Others you often use:	

3. Please estimate the relative amount of time your computer is used for business versus recreational purposes:

% business % recreation

4. CP/M users: Do you have access to the CP/M users group library on media your computer can use?

( ) Yes. I can get these programs...
- ( ) direct from the CP/M Users Group
- ( ) from a computer club in my area
- ( ) from a friend
- ( ) from:

( ) No, the users group doesn't have my kind of diskettes.

If not, would you like access to these programs?
- ( ) No
- ( ) Not sure. I'd like to see what the library has.
- ( ) Yes, but only on my kind of diskettes.
- ( ) Yes, I'd even take them on cassettes in a format suitable for loading onto my disk (assuming the price is reasonable).

(Continued on other side)

PROTEUS QUESTIONNAIRE #1 CONT.

5. Would you buy cassettes containing public-domain library programs for cassette systems? ( ) Yes. ( ) No. ( ) Perhaps.

If so, what do you consider a reasonable price for a C-60 cassette from the library, including postage? From $__ to $__.

6. What kinds of programs would you like to see in a library? Indicate by showing the percentage of Proteus library space you personally would like devoted to each topic:

__% Games
__% Business applications (accounting, analysis, etc.)
__% Programming systems and aids (compilers, debuggers, etc.)
__% Statistical, engineering, scientific
__% System utilities (file manipulations, device drivers, etc.)
__%
__%
__%
__%
__%

7. What kinds of articles would you like to see in Proteus News?

8. What other services would you like Proteus to perform?

9. What didn't you like about our performance as "Solus" in the past year?

10. What did you like most about Solus?

11. What do you think about the current issue of Proteus/News?

12. Would you participate in a software-swap meeting of Proteus at the 4th West Coast Computer Faire, May 11-13, in San Francisco? It would be arranged so that you could either mail your software ahead of time, or drop it off when you arrive at the show, and then pick up copies of all submitted software at the end of the show.

( ) No, I don't have any original software to contribute.
( ) Yes, but by mail only.
( ) Yes, I'll attend and can drop off my software there.

What else would you like to have happen at such a meeting?

( ) A tutorial course on the following topic:
( ) A panel discussion by Proteus members on:
( ) A presentation by Processor Tech on:
( ) Other:

P R O T E U S O R D E R F O R M

PROTEUS LIBRARY CASSETTES (PUBLIC-DOMAIN PROGRAMS)
An annotated catalog of each cassette is published in Vol. 2, Number 1, and subsequent issues.
Price: Each cassette is $8 with a donated program, or $18 without an acceptable donated program. See the Copyright Statement on the other side if you are donating a program.
To order: Circle the item codes.
Items ordered: C1 C2 C3 C4 C5
Number of cassettes @ $18 each (without donation):
Number of cassettes @ $ 8 each (with donation):
(One donated program must be submitted for each tape purchased at the $8 price.)
TOTAL PRICE OF CASSETTES FROM LIBRARY: $ ____

HELIOS LIBRARY DISKS (PUBLIC-DOMAIN)
An annotated catalog of diskette H1 is in Solus News (the precursor of Proteus/News) Vol. 1, Number 6. Diskette H2 will be described in the next issue of Proteus/News. It will contain the source code for the SLAC PASCAL system and miscellaneous donated programs.
Price: $25 each diskette without acceptable donated program, or $10 with an acceptable donated program. See Copyright Statement for donated programs, on reverse side.
To order: Circle the diskette item number.
Helios diskettes ordered: H1 H2
Number of diskettes @ $25 each:
Number of diskettes @ $10 each:
(An acceptable program or data file must be submitted for each diskette ordered at $10 price.)
TOTAL PRICE OF DISKETTES ORDERED: $ ____

COMMERCIAL PROGRAMS (PROPRIETARY, NOT PUBLIC-DOMAIN)
G/2 Extended BASIC cassette. (List $49.95) __ @ $42.46 ____
G/2 Beat the House cassette. (List $14.95) __ @ $12.71 ____
G/2 Clinic cassette. (List $14.95) __ @ $12.71 ____
G/2 Outwit cassette. (List $14.95) __ @ $12.71 ____ $ ____

TOTAL PRICE OF ALL PROGRAMS ORDERED: SUB-TOTAL $ ____

Calif. residents sales tax (6% of previous line) TAX $ ____

PERIODICALS
Current subscription to Proteus/News (1979) ____
In the U.S.A., $12 per calendar year.
Foreign addresses, $15 per calendar year.
Subscriptions expire at end of calendar year (Dec 31).
Mid-year subscribers will receive back-issues of current year. Subscription includes membership in Proteus, required to purchase library programs or obtain discounts on commercial products.
Back-issues of 1977 (Vol. 0, Solus News) @ $2/set ____
Back-issues of 1978 (Vol. 1, Solus News) @ $10/set ____

(PERIODICALS ARE NOT SUBJECT TO CALIF. SALES TAX) $ ____

TOTAL AMOUNT OF PURCHASE TOTAL$ ____
(MAKE PAYABLE TO PROTEUS. US FUNDS ONLY PLEASE.)

→ O V E R ←

PROTEUS ORDER FORM (CONT.)

LIMITED WARRANTY ON PROTEUS LIBRARY PROGRAMS

The programs included in the Proteus Library have been donated by members as original works or from the public domain. Proteus has screened the programs to check that they work, but Proteus does not warrant them to be correct. Some of the programs may bear restrictions on their use and so indicate in the text of the program. Media that cannot be read by the purchaser's appropriate equipment may be returned within 30 days from date shipped, for replacement or repair.

Proteus assumes no liability to anyone with respect to any loss, damage, or liability resulting directly or indirectly from the use of these products, including but not limited to interruption of service, loss of income, or consequential damages arising from the use of the product.

COPYRIGHT STATEMENT (REQUIRED WHEN FILES ARE DONATED TO LIBRARY)

If you are donating a program or data file to Proteus for a reduced price on purchase of a Proteus product, please complete the following statement:

As donor of the computer file(s) or program(s) named below, I certify that no copyrighted work is contained therein, other than my own. Furthermore, if my submission is accepted by Proteus (in whole or in part), I hereby transfer, assign, or otherwise convey all copyright ownership to Proteus.

Name of files or programs donated: _____

Donor's signature: _____

Date: _____

MAIL COMPLETED ORDER FORM TO:

PROTEUS
1690 WOODSIDE ROAD, SUITE 219
REDWOOD CITY, CALIFORNIA 94061
U.S.A.

ALL PRICES INCLUDE POSTAGE BY UPS OR SURFACE MAIL. IF AIR SHIPMENT IS DESIRED, LET US KNOW AND WE WILL NOTIFY YOU OF ADDITIONAL CHARGES.

IF PAYMENT ACCOMPANIES ORDER, WE PAY POSTAGE.
IF C.O.D. IS DESIRED, WE WILL ADD SHIPPING AND C.O.D. CHARGES.
C.O.D. NOT AVAILABLE TO FOREIGN ADDRESSES.

U.S. FUNDS ONLY, PLEASE.

LATE NEWS! >>>> PTC SOURCE CODE FOR SALE <<<< LATE NEWS!

For the past 6 months, we've been trying to get in touch with someone at Processor Tech who has authority to negotiate sale of their source code, and finally we succeeded! The source code for PTDOS (and its subordinate commands), ECBASIC, EDBASIC, ALS-8, and in fact for any PTC program that was produced in-house, is for sale by PTC for an amount in 5 figures, depending upon exactly what we buy. What we will try to do is put together enough buyers to collectively purchase what we want. If we don't act fast, the Corporation may be forced into bankruptcy court by its creditors and we will lose the opportunity to negotiate this purchase.

The purchase agreement will probably state that buyers can use the source code in any way (commercially or otherwise), but they cannot resell or otherwise disclose the source to anyone else. The sale is non-exlusive, that is, PTC is selling this to anyone who wants it, but no buyer can turn around and sell or give away the source, just object code. I'll try to negotiate a time limit on the non-disclosure clause, so that after some time the package could go into public domain.

The agreement will include the right to reproduce the user's manual, and hopefully a machine-readable copy of it.

To keep the price within reach of the greatest number of members, we are splitting the software into a few packages. We haven't seen the programs yet, so we have no idea of the amount of code that we will have to reproduce. Hence, we have separated the cost for reproduction and mailing from the purchase of the source code itself. After we see what we have, we will let you know the reproduction cost, but the rates will be our usual--that is, $0.15 per page, $8 per cassette, $10 per diskette.

So, here are the packages and prices, with their Proteus item numbers.

Proteus item P10:
Source code and object code for the cassette versions of BASIC/5 language, FOCAL language, Software Package #1 (precursor of ALS-8), and ASSM (cassette-to-cassette assembler). Price $45 plus reproduction costs.

Proteus item P11:
Source code and object code for GAMEPAC 1, GAMEPAC 2, and TREK-80. Price $15 plus reproduction costs.

Proteus item P12:
Source code and object code for Extended Cassette BASIC, Extended Disk BASIC, Optional Precision Extended Disk BASIC, and possibly the new Level I Business BASIC (Disk) which allows program chaining with passing of data in COMMON area and other enhancements for modular business systems. Price $90 plus reproduction costs.

Proteus item P13:
Source code and object code for Extended Cassette BASIC only. Price $45 plus reproduction costs.

Proteus item P14:
Source code and object code for ALS-8, the Assembly Language System consisting of an assembler (memory-to-memory), video editor, and 8080 simulator. Price $45 plus reproduction costs.

Proteus item P15:
Source code for PTDOS, the Processor Technology Disk Operating System. We presume this will be version 1.5, although it might also include the version 2.0 which was under development when PTC went out of business. It will contain all of the source modules for the command and device files on the distributed disk except the source of the Extended Disk BASIC, which is a separate item listed above. Price $90 plus reproduction costs.

Proteus item P16:
Source code and object code for Sol diagnostic and test programs that were designed to aid the technician in trouble-shooting a malfunctioning Sol system. Includes 16KRA, 16KDT, 32KRA, 48K (all preceding are memory tests), DMARD and DMAWR Helios DMA tests, SOLT Sol Diagnostic Test program, SOL-B Sol burn-in test program, DOST Sol System III automatic tests, DISKT Helios Disk Diagnostic Program, DISKCHK Helios file checker, SIMU (Simu-Ciser) a program to exercise the PerSci floppy disk drive during alignment of the drive, PSRC1 thru PSRC3 ParaSol debugger programs for the ParaSol board. Price $25 plus reproduction costs.

User's manuals for all of the acquired programs will be made available as Proteus document items (prefix "D" in item number), both individually and in groups corresponding to the above item groupings. In addition, we plan to produce a "Book of Sol" in several volumes, containing portions of PTC technical manuals that have not been generally available. As part of the deal with PTC, we plan to acquire the right to reproduce these manuals, and part of the purchase prices you pay will go toward this.

To purchase your copy of the items above, send us your order immediately. Payment of the above mentioned prices should accompany your order. Once the programs are acquired, we'll let you know how much extra it will cost to reproduce them, but the rates will be as mentioned above. If we can't put together enough money to make the deal, we'll refund your money. In your order, tell us what media you want the programs on: paper listing only, cassette, Helios disk. We'll also make the programs available on standard 8" soft-sectored CP/M diskette and NorthStar diskette, but we can't promise swift delivery of this form because of the time consuming process of transferring the data to the other media.

Acquisition of these programs will help prolong the useful lives of our systems. I hope we have adequate participation by the members. This is the most significant event in the history of Proteus.

Sincerely yours,

Stan

Stanley M. Sokolow

PROTEUS / NEWS

AN INDEPENDENT NEWSLETTER FOR OWNERS AND USERS OF PROCESSOR TECHNOLOGY CORPORATION COMPUTERS

FORMERLY SOLUS NEWS

1983
V. 6, #4

PUBLISHED BY PROTEUS, 1690 WOODSIDE ROAD, SUITE 219, REDWOOD CITY, CA 94061, USA

SINGLE ISSUE ...
$7.50 (US)
$9.50 (Foreign)

A F A R E W E L L N O T E

This issue marks the end of an era in the history of the Sol computer. We started publication of this newsletter in August, 1977, after the first formal meeting of the Sol user's group. This issue, 7 years later, is the last one we are going to print. In its place, we will continue to send announcements of general interest to Sol owners, but only at irregular intervals and only when we have sufficient material to warrant it.

The original Sol user's group, called Solus back then, began with about 30 owners. Over the years, our numbers swelled to a high point of about 1200 and gradually declined to the present level of about 200 members. Of course, there were many more Sols made, but only some went to personal users who were interested in a user's group.

In the early days, we desperately needed each other for information and help. After Processor Technology went out of business, I felt we still needed each other more than ever. But today, most people have other options for handling their computing needs. A whole new generation of computers has come and will soon pass into obsolescence. Mass market computers are available with more computing power than our Sols.

The microcomputer world has moved on beyond the Sol, but the excellent design of the Sol has stood the test of time for those who continue to use it. By avoiding unusual components and by incorporating the S-100 bus, the designers created a rugged computer which will continue to be a servicable machine for many years to come.

It has been a great educational experience for me to be the focal point of information about the Sol. I started with a background in computer programming from my college years, but very little electronic knowledge and virtually none about digital logic and computer hardware. By reading Adam Osborne's original microcomputer books, articles in Byte magazine, Don Lancaster's TTL cookbook, etc., and by absorbing the comments of the skilled Sol owners who contributed material to the newletter, I eventually could understand the Sol technical manual.

Out of necessity, I learned how to maintain my hardware that was orphaned. Just recently, I purchased my own oscilloscope for maintainance of my PerSci disk drives. I can read the schematics and trace the signals down to the faulty gate or transistor. I've come a long way. I look at the oscilloscope as another educational tool that will carry me into the future of computing. We all have a long way yet to go.

If you haven't yet taken the plunge into understanding the insides of your computer, but you consider computers your hobby, think about making a move in that direction. It's a good feeling to know that, as long as you can find the parts, you can be self-sufficient with your computer, even if it is an orphan.

Some day, our Sols will be antiques. In fact, when asked what kind of computer I have, I usually tell people "Oh, you never heard of it; it's an antique made before the guys at Apple sold their first Apple II." Let's never lose sight of this fact. We've got the equivalent of the antique classic car.

I should give special thanks to a few people whose help made this journey last as long as it has. First, Bill Burns deserves special mention as the founder of Solus. It was his impetus that pulled together the Sol owners at the Homebrew Computer Club to form the first Solus meeting and subsequently to extend it to the world-wide Sol owners' group.

Second, the designers and entrpeneurs who started Processor Technology Corporation: Gary Ingram, Bob Marsh, Lee Felsenstein, and any others. I'm sure I speak for all Sol owners when I say that we were sorry to see Processor Tech go under, and we still have dreams of what it may have become, if only ....

Third, Les Solomon, technical editor of Popular Electronics (now known as Computers and Electronics) magazine. Those who don't know the story should read the historical material in the Encyclopedia Processor Technica , volume 1. The Sol is named after Les, and it owes a large part of its success to his early support in the magazine.

Fourth, I want to thank the major contributors of articles and time: Joe Maguire, Father Thomas McGahee, and others. You helped to make Solus News and Proteus News one of the best users' group newsletters, according to the many positive comments I received in letters from readers.

And certainly I want to thank all of you members who are reading this issue. Without your diehard support, many people would have been left stranded without any help after investing a large amount of their money in a Sol system. The continued membership support made this all possible.

One person deserves special mention because he is part of your future with the Sol: Bob Hogg. Bob is a retired engineer who has an uncommon level of expertise in digital electronics and in the Sol in particular. He is the proprietor of Micro Complex, the firm that is still supporting Sol owners with service and new products: the 64/80 column video upgrade, the dual personality module, his NorthStar work-alike improved disk controller (Phase Lock II), and so on. If you still love your Sol and want to keep it alive and growing, look to Bob for the help you need.

One last word: I'm quite busy now and not able to spend time with Proteus, but I'll always be reachable by phone. Here's a tip to tuck away with your Sol documentation. I'm a California licensed dentist, an orthodontist to be specific, so you'll always be able to track me down, even in the unlikely event that I move away. The California State Board of Dental Examiners (in Sacramento, California) will always know my business address. If you are absolutely stuck and can't seem to find anyone to help with a Sol problem, I'll be glad to give you any advice I can muster up out of my memory or to point you toward another contact.

So, best regards to you, especially the old-timers who've been with us since the start. It's been good to know ya'.

Sincerely,

Stan Sokolow

==========

McZOL

THE Z-80 UPGRADE FOR SOL FROM MICROCOMPLEX

Bob Hogg has been working on his Z-80 upgrade for a long time. It wasn't easy to get the Z-80 to work reliably with the Sol and its most common peripheral equipment. Now you can obtain a Z-80 upgrade from MicroComplex, but you should read this article before you buy one.

The Z-80 upgrade is a piggy back board that plugs into the main Sol PC board, just like the 64/80 column video board does. It takes up no slots. Since Bob designed the two upgrades to work together, you can install both in the same Sol and have a computer that will run your existing software as well as the new software designed for the Z-80 and 24 x 80 video screen.

The Z-80 replaces your present 8080 microprocessor, but since it has an expanded version of the 8080 instruction set, the Z-80 can run both 8080 and Z-80 programs. Bob has designed the board to run at 2 or 4 MHz clock rate, or optionally at higher clock frequencies. The faster the clock, the faster the computer.

The Z-80 upgrade can be used with a Sol that does not have the video upgrade, and instructions are given in the installation guide. I installed my Z-80 piggy-back in a Sol that has the video upgrade, without removing the Sol-PC board. This requires some care to be sure you don't bend the pins that insert in the Sol's sockets, but it doesn't require uncommon skill. It's smaller and easier to insert than the video board.

You need to remove the Sol covers, keyboard, and several IC's. A few IC's have to be removed, certain pins bent outward to avoid the socket, and re-inserted. A few soldered wires must be added. The keyboard needs to have a couple of passive components (R & C) replaced to be compatible with both the original and the faster clock speed.

You may have a problem with your S-100 memory boards. Processor Tech dynamic memory boards were designed only to work with a 2 MHz clock and they absolutely won't go at 4 MHz. This is also true of many other older dynamic memory. I have static memory in the modified Sol, with memory chips rated at 300 ns. These work fine. If you need new memory, I've seen boards with the newer CMOS static chips advertised for about $300 for a 64K board. Check the back of Byte magazine for ads from Jade, Advanced Computer Products, etc.

You may also have trouble with your disk controller. Bob has been beating his head against the oscilloscope trying to fix the design to work with the NorthStar double density disk controller. NorthStar took shortcuts that make their board too sensitive to transients on the bus, and the Z-80 board just overwhelms the disk controller. Bob's Phase Lock II controller (which I dubbed the McFloppy) is a more robust design than NorthStar's, and it is compatible with NorthStar's format, so that's an escape. But it adds to your cost if you must go that route. The single density NorthStar controller works fine with the Z-80 upgrade.

I haven't fully checked out the nature of the problem yet, but when I tried to bootload my Helios disk system using PTDOS in the Z-80 modified Sol (which I'll call my McZol from now on), the bootload failed to come up. A modification to the Helios controller is needed for compatibility (two jumpers), but I made that and still no luck bootloading. I haven't had time to try other controllers or to look at possible problems with my installation.

So, if you have a Helios, procede with caution. I know that PTDOS has some timing loops in software, and this may be the problem. Bob Hogg tells me that at least one other McZol is running a Helios with Lifeboat CP/M, so it sounds to me like a software problem, not hardware. The fix may only take modification of a single byte to change a timing loop counter, but I haven't looked into the details. Bob's hardware allows a program to switch the clock speed from 2 to 4 MHz, so I can try booting up a slow speed, but I haven't done that yet.

When you order your McZol upgrade, be sure to tell Bob what kind of IC sockets you have. The Texas Instruments low-profile (edge-wiped contacts for the IC pins rather than surface wiped contacts) require a different setting of the pin height when the upgrade is assembled by Bob.

Bob has tried his Phase Lock II controller in a McZol at speeds up to 10 MHz with no problem. (The video ram and proms have to be replaced with newer ones that are faster than the ones in a standard Sol, of course.) Z-80 chips that run at that speed over the full commercial temperature range are not available yet, but this fact gives you the assurance that the hardware can handle more than you are likely to give it.

I ran some simple BASIC programs with Extended Cassette BASIC, and yep, they ran twice as fast at 4 MHz as they do at 2 MHz. But, when I asked Solos for a memory dump, I noticed that it didn't run as fast as I expected. Then I remembered! The Sol PC introduces wait states when running programs in the Solos ROM.

The wait states were designed into the Sol to let it use slow EPROMs at the normal 2MHz clock speed and are not needed if you use faster ROMs. But at 4 MHz, you need the wait states again; the installation instructions tell you how to re-introduce the wait states if they have been jumpered out of your Sol. The wait states for I/O can be removed (the instructions tell you how). The wait state circuitry on the Sol only slows down programs when they access the on-board Solos ROM, not when they access S-100 memory. Your programs will of course reside in S-100 memory, so you will take full advantage of the new speed.

You can order the Z-80 upgrade from MicroComplex, 25651 Minos Street, Mission Viejo, CA 92691, telephone (714) 770-2168. Price: $150 for the basic board fully assembled and tested, but you install it; optional extra features available; installation available at modest additional cost. (Bob is a night person. The best time to call is in the afternoon or evening, Pacific time.)

Unclassified Ad

FOR SALE - Operational computer system consisting of:
SOL-20 Rev E. 56K User RAM from various manufacturers. Northstar single-density controller with dual drives. Sanyo 9" b-w monitor. Assorted software including: EC BASIC, EDIT modified with text processor, ALS-8, DEBUG, ASSM, Gamepac 1 and 2, Trek-80, assorted machine language and BASIC games and utilities. Programs that had cassette access will also be supplied with disk drivers. All pertinent manuals plus ALS-8 Users group letters and notes. JADE Bus Probe.

Will sell entire system for $500 or best offer received one month from the publication date. Will not sell parts individually.

FR. THOMAS MCGAHEE
Don Bosco Technical High School
202 UNION AVENUE, PATERSON, NEW JERSEY 07502
Telephone: (201) 278-8800

Dear Stan and SOL Brothers,

Enclosed is the manual and complete source listing for the **SUPER-USER AREA** routines that Rick Downs mentioned in his letter to PROTEUS. You have my permission to reprint all of this in its entirety. I am sure that many SOL users will find things of interest in the program. As usual, it is heavily commented so that others may make changes more readily.

Although this program states that it is for LIFEBOAT CP/M version 2.X, the actual interface is consistent with ALL versions of CP/M that I have tested it with. I have, for instance, used it with the HELIOS CP/M 1.4, MICROPOLIS CP/M version 2.2, NORTHSTAR CP/M version 2.2, as well as with version 2.2 CP/M for TARBELL, MICROMATION, VERSAFLOPPY and others. There is one section marked with <;$$$> that will have to be changed for each specific system, but that is a minor undertaking. Just look up what your CURRENT system equates are, and put those in. The program has lots of goodies in it, and so it eats up almost 512 bytes. In systems where the BIOS does not have sufficient room for all this, it can be kept in a separate file and relocated into the top of memory and then patched into the BIOS using a short program. It can be used as-is by any LIFEBOAT version of CP/M that supports a separate USER AREA.

I originally wrote this program to specifications supplied by Rick Downs. Because it was a custom program, I charged him $65 to help cover the time I spent writing it. I am now releasing it through PROTEUS in the hope that other SOL users may benefit from it. You guys out there who find it useful can thank Rick Downs for getting me to write it in the first place. I have added some additions and improvements to the original program specifications, and as far as I can determine the program is bug-free.

I would like to remind our readers that we here at DON BOSCO TECHNICAL HIGH offer a three-year course in computer programming as well as a three-year electronics course in which we teach computer fundamentals and interfacing. Any donations of computer equipment (especially SOLS) is greatly appreciated. You would not believe how much use we get out of our equipment! Our machines are running from 8 AM to 10 PM almost every day of the week. We need some additional

equipment to handle replacement of machines while they are being repaired. In addition, we have several projects underway that require the use of a computer as a controller. For example, current projects are using the computers to control ROBOTS, our own homemade video games, a LASER beam that writes on the wall, computerized music synthesis, a vector-beam graphics system, and a model computer language. We keep pretty busy around here.

When sending any donated equipment, include an itemized list of your original cost. I will see to it that you receive an itemized acknowledgement for your tax records.

Some people like Dr. Richard Black and Jim Blackwood have already donated SOL systems to us, and these systems are now in daily use here at DBT.

We can even use certain so-called BASKET CASES as spare-parts sources. (Besides taking care of our own SOLs here at DBT, I also help other SOL users get their systems modified and repaired). Almost anything related to SOLs is of use to us.

I will try to get some more articles out in the near future, as time permits.

Sincerely yours,

Fr. Thomas McGahee

P.S. We have an IBM PC here that I have been doing a lot of programming on, but it is still not as useful as our SOLs at this time because I have much to learn about its innards yet. I have also been heavily into DBASE II programming the past two years, and have developed a lot of neat programs and techniques that I eventually want to share.

WRITTEN BY
FR. THOMAS MCGAHEE
DON BOSCO TECHNICAL HIGH SCHOOL
202 UNION AVENUE
PATERSON, NEW JERSEY 07502
(201) 595-8800

CUSTOMIZING USER.ASM TO MEET YOUR NEEDS

USER.ASM is provided in SOURCE CODE so you can easily make changes to it to customize it to your needs. Some sophisticated users may wish to actually change some of the program code, but most persons will find that they can effect the necessary customization by simply setting a few EQUATE statements. Be aware of the fact that, depending on the options you select, up to 511 of the 512 bytes available in the USER area may be used. If you make changes to the actual PROGRAM (as opposed to just choosing options with the EQUATE statements), then make sure that you examine the value of <LFTOVR> at the VERY END of the USER.PRN listing the assembler generates. Recall that the assembler uses two-s complement notation. If the value of <LFTOVR> (The amount of space LEFT-OVER) starts off with an "F", then you have a NEGATIVE amount... meaning you have exceeded the 512 byte limit.

Using whatever editing facilities you have available, load in the USER.ASM file. In systems with very little memory, you may wish to just append the first 300 lines, as the entire program is very large. This is due to the fact that I have heavily commented the program so people can figure out what I am doing. All lines that might possibly require customization are marked with <;***> or <;$$$>. Thus you may use the search facilty of your editor to search for <;***> or <;$$$> to locate all such lines.

I will now outline each such line and give a brief description of its use and possible options. To make things easier, I will give them in the order they actually appear within the listing.

MSIZE Set for the amount of contiguous memory in your system.

SCRN16 Set <TRUE> if using 64x16 screen, or <FALSE> if using the MICROCOMPLEX 80x24 board by BOB HOGG. Setting this will cause automatic generation of the SYSTEM RAM addresses as required.

PAGEL Set for the number of actually printed lines desired per page of printed text. A normal sheet of paper is 66 lines long. I suggest setting this to 55, as this allows a decent margin at the top and bottom of the sheet.

NEWPAGE Set this <TRUE> if you want an automatic FORMFEED generated whenever you use LOAD/F, LOAD/S, or LOAD/^ to setup the printer.

SOLOS Set this for the starting address of the SOLOS ROM in your system. <0C000H> or <0F000H> are usually used.

CONSIN Set to the device you want used for console input. May be KBD, SERIN, PARLIN, or CSTMIN.

**** NOTE **** KBD=SOL KEYBOARD, SERIN=SOL SERIAL INPUT PORT, PARLIN=SOL PARALLEL INPUT PORT, CSTMIN=USER-DEFINED CUSTOM INPUT PORT WHOSE ADDRESS IS AT SOLOS <UIPRT>.

CONSOUT Set to the device you want used for console output. May be VDM, SEROUT, PARLOUT, or CSTMOUT.

**** NOTE **** VDM=SOL VIDEO SCREEN, SEROUT=SOL SERIAL OUTPUT PORT, PARLOUT=SOL PARALLEL OUTPUT PORT, CSTMOUT=USER-DEFINED CUSTOM OUTPUT PORT WHOSE ADDRESS IS AT SOLOS <UOPRT>.

LISTOUT Set to the device you want used as your LIST device (PRINTER). May be VDM, SEROUT, PARLOUT, or CSTMOUT.

LISTIN Set to the device used to read from LIST device. This is used for handshaking purposes, and can generally be set as follows: <LISTIN EQU LISTOUT>. This is the default setting. You may also explicitly set it to SERIN, PARLIN, or CSTMIN.

RDRIN Set to the device you want used as the READER. It may be SERIN, PARLIN, or CSTMIN. I suggest that if you are using PARLIN for the LISTIN device, then use SERIN as the RDRIN device. If using SERIN as the LISTIN device, then use PARLIN as the RDRIN device. There is nothing wrong, of course in having LISTIN and RDRIN assigned to the same device. In many systems it is useful to use the READER for reading in from MODEMS.

PNCHOUT Set to the device you want used as the PUNCH. It may be SEROUT, PARLOUT, or CSTMOUT. I suggest that if you are using PARLOUT for the LISTOUT device, then use SEROUT as the PNCHOUT device. If using SEROUT as the LISTOUT device, then use PARLOUT as the PNCHOUT device. There is nothing wrong, of course in having LISTOUT and PNCHOUT assigned to the same device. In many systems it is useful to use the PUNCH for sending to MODEMS.

NONEH Set <TRUE> only if your LIST device (PRINTER) requires no handshaking. Printers using the PARALLEL port do not generally require handshaking, as this is done in hardware by the parallel port driver in SOLOS. Also, certain non-intelligent printers such as the venerable TTY do not require any handshaking.

ETXH Set <TRUE> only if your LIST device (PRINTER) is set up to handle ETX/ACK handshaking. For example, DIABLO and most DAISY WHEEL and THIMBLE printers can generate and use this protocol. Some such printers need to have the protocol explicitly set by switch somewhere, whereas others use this protocol without need for any switch settings. See your printer manual for details.

XONH Set <TRUE> only if your LIST device (PRINTER) is set up to handle XON/XOFF or DC1/DC3 or CTL-Q/CTL-S handshaking. Most DOT MATRIX printers use this protocol, and many DAISY-WHEEL and THIMBLE printers can select this protocol as an option. See your printer manual for details.

BUFFR Set this to the length of your printer's buffer. This is only needed for ETX/ACK handshaking. The default value of 80 is for DIABLO printers and most DAISY WHEEL and THIMBLE printers. If not sure about your printer's buffer size, leave it at 80... it will still print at its maximum speed! Buffer size is used to determine how often ETX/ACK handshaking must be done, and that's all. Maximum value for <BUFFR> is 255.

BANK Set <TRUE> only if you are using a bank-select scheme such as my SUPER-PHANTOM to allow RAM to overlay SOLOS ROM/RAM. This allows CP/M systems greater than 48K to be generated and run. Please note that the entire USER area must reside outside the SOLOS ROM/RAM address space for the overlay scheme to work properly. In addition, unless the CPM system has been patched to change the way stacks are assigned, then the entire CPM operating system must reside outside the SOLOS ROM/RAM area. If this is not done, then stack control may be lost when banks are switched.

BNKPRT Set to value of port used for bank-selecting. In my SUPER-PHANTOM technique, OFFH is used.

SOLON Set to value to be sent to bank-select port to cause SOLOS ROM/RAM to be ACTIVATED.

SOLOFF Set to value to be sent to bank-select port to cause SOLOS ROM/RAM to be DE-ACTIVATED and OVERLAYED by RAM.

That takes care of the actual system EQUATES. There is also a set of INITIALIZED values that determine the DEFAULT conditions when the USER area is first loaded from disk. The following detail these locations and their default values:

ATOGGLE <00>=OFF <01>=ON. When ON, the video screen will scroll up a maximum of one full page and then stop. Hitting any key will cause it to start scrolling the next full page. It is recommended that this be set to <00>. Please note that when using the SHEET-PAPER option and doing a listing, this should be toggled OFF, as the SHEET-PAPER option will also suspend listing until a key is hit, and you may get confused as to what hitting the key will do!

LTOGGLE Set this to <01> ONLY if you are setting <PAPER> to <'F'>, and you want perforation skipping (Automatic Formfeed every <PAGEL> number of lines). Otherwise set it to <00>.

PTOGGLE <00>=OFF <01>=ON. This determines whether or not the PRINTER is active. When active, everything that appears on the CONSOLE OUTPUT device will be echoed on the PRINTER. NOTE: This feature is INDEPENDENT of the CPM CTL/P function. This is toggled using the LOAD/P sequence. Don't have CTL/P and LOAD/P BOTH ACTIVE, or the printer will double-print text.

PAGE <PAGEL>. Leave this set to <PAGEL>. This starts the program logic off so that it thinks it is at the top of a page. NOTE: You still have to set your printer Top-Of-Form control if you have one.

PAPER <'S'>=SHEET <'F'>=FAN-FOLD. Set this to the kind of paper you generally use. LOAD/S and LOAD/F and LOAD/^ can be used to change the value under program control later.

*** The remaining INITIALIZED data items should be left as-is.

USING THE NEW FEATURES

MODE-SELECT This is converted to a CONTROL/C. Since CONTROL/C is used so often, it is nice to have a single-key that can be used in its place. The MODE-SELECT key was chosen because it is so large and conveniently located.

LEFT-ARROW The SOL LEFT-ARROW CURSOR KEY is converted to a BACKSPACE, since the SOL has no backspace key. Note that in CPM a backspace will cause the cursor to back-up and ERASE the last character. (DELETE and LEFT-ARROW are equivalent functions now, as both act like backspace).

DELETE The DELETE KEY is converted to a BACKSPACE. Note that in CPM a backspace will cause the cursor to back-up and ERASE the last character. (DELETE and LEFT-ARROW are equivalent functions now, as both act like backspace).

NOTE The output routines have been designed so that a BACKSPACE will be properly handled. Also, multiple carriage-returns no longer cause stuff on the screen to be erased.

THE LOAD KEY HAS BEEN IMPLEMENTED AS A KIND OF "ESCAPE" FUNCTION. THE FOLLOWING LOAD SEQUENCES HAVE MEANING:

LOAD/A Toggle the AUTO-DISPLAY feature. When this is ON, it causes the screen to stop scrolling after one full page. To resume scrolling of next page, hit any key except LOAD.

NOTE Any LOAD sequence that produces a TOGGLE-ACTION will result in a reverse-video <1> or <0> being displayed to indicate whether it is now **ON** or **OFF**. The actual LOAD sequence itself is always done without any echoing of characters on the screen. Please note that **the program will convert LOWER CASE to UPPER CASE automatically within the LOAD SEQUENCE.** This is a convenience feature so you don't have to worry about what case you're in.

LOAD/0 - LOAD/9 Sets the speed at which scrolling takes place. <0>=FASTEST <9>=SLOWEST.

LOAD/P Toggles the PRINTER ON and OFF. Note that this is independent of the CPM CONTROL/P function. If your implementation of CPM supports the CONTROL/P function, then do not activate it and this at the same time, or else double-printing will occur. Unlike CONTROL/P, LOAD/P may be invoked at any time, even in the middle of a listing!

LOAD/S Unconditionally sets the printer up to handle SHEET-PAPER. If <NEWPAGE> was set <TRUE> in the assembly listing, then an automatic FORMFEED will also be generated at this time. In any case, the <PAGE> count is set to <PAGEL> to indicate that we are at the top of a new page. When SHEET PAPER is being used, the PRINTER will automatically receive a FORMFEED signal at the end of each page. It will then suspend further printing until any key is hit on the CONSOLE INPUT device (usually the keyboard).

LOAD/F Unconditionally sets the printer up to handle FAN-FOLD PAPER with continuous printing. If <NEWPAGE> was set <TRUE> in the assembly listing, then an automatic FORMFEED will also be generated at this time. This mode is very handy when printing MAILING LABELS and the like, because it will NOT generate an automatic FORMFEED at the end of a page.

LOAD/^ Unconditionally sets the printer up to handle FAN-FOLD PAPER with AUTOMATIC FORMFEED generation at the end of each PAGE. (That is, it will skip over the perforations in the paper, making for a much neater listing). If <NEWPAGE> was set <TRUE> in the assembly listing, then an automatic FORMFEED will also be generated at this time. In any case, the <PAGE> count is set to <PAGEL> to indicate that we are at the top of a new page.

LOAD/MODE-SELECT Causes a JUMP to SOLOS. (This is an entry to the SOLOS <RETRN> entry point at SOLOS+4). Think of it as being like a SYSTEMS RESET, except that the screen is not erased, and the SOLOS SYSTEMS RAM is not cleared. Should you prefer the screen erasure and clearing action, change the assembly listing after the label <IN4:> from <JMP SOLOS+4> to <JMP SOLOS>.

LOAD/LOAD This allows you to temporarily suspend screen operations. Whenever you hit the first LOAD, the display will freeze. When you hit the second LOAD, it will continue. This is much more convenient that using the CPM CONTROL/S feature. Since printing is tied in with the display, this is also a way of temporarily suspending printing.

ANY OTHER LOAD SEQUENCE NOT OUTLINED ABOVE WILL ACT LIKE THE LOAD/LOAD SEQUENCE AND MERELY PAUSE THE DISPLAY.

One last NOTE: I have incorporated the key conversion and LOAD sequence interpreter into the CONSOLE STATUS routine instead of the CONSOLE INPUT ROUTINE. This allows all features to be activated AT ANY TIME. In addition, whenever a LOAD sequence is entered, the return to CPM is done in such a way that CPM does not know that anything happened. Since a TOGGLE can be either ON or OFF, I have included a routine that displays a REVERSE-VIDEO <0> for OFF and <1> for ON. CPM is totally unaware that these REVERSE-VIDEO codes have been displayed. They are simply there for your convenience.

```
;*****************************************************************
;*****************************************************************
;*****       CUSTOM USER AREA FOR LIFEBOAT CP/M 2.X          *****
;*****************************************************************
;*****  MAY SERVE AS THE I/O SECTION IN NON-LIFEBOAT CP/M   *****
;***** BY JUST CHANGING THE EQUATES MARKED WITH $$$ IN THE  *****
;***** COMMENTS.                                            *****
;*****************************************************************
;*****  CUSTOMIZED FOR USE WITH SOL-20 AND MOST PRINTERS    *****
;*****************************************************************
;*****************************************************************
;*****             WRITTEN BY FR. THOMAS MCGAHEE             *****
;*****               DON BOSCO TECHNICAL HIGH                *****
;*****                     202 UNION AVE.                    *****
;*****                  PATERSON, NJ  07502                  *****
;*****                    (201) 595-8800                     *****
;*****************************************************************
;*****************************************************************
;*****               LAST REVISED JUNE 4, 1984               *****
;*****************************************************************
;*****************************************************************
;*****           MODIFIED TO ALLOW NULLS AFTER CR            *****
;*****     GENERATES FORMFEED VIA CALCULATED LINEFEEDS       *****
;*****************************************************************
;*****************************************************************
;***** IF YOU FIND THIS SOFTWARE USEFUL AND WISH TO MAKE A *****
;***** $5 DONATION TO HELP ME CONTINUE PRODUCING SUCH STUFF*****
;***** I WOULD APPRECIATE SUCH A KIND GESTURE OF SUPPORT.  *****
;*****************************************************************
;*****************************************************************
;***** IF YOU HAVE AN OLD SOL YOU NO LONGER USE, WE HERE AT*****
;***** DON BOSCO TECH WOULD LOVE TO GIVE IT A NEW HOME     *****
;***** WHERE IT WILL GET LOTS OF USE AND TENDER LOVING CARE*****
;***** <DON BOSCO TECH HAS A THREE-YEAR COMPUTER COURSE IN *****
;***** WHICH WE GIVE STUDENTS OVER 700 HOURS OF INSTRUCTION*****
;***** AND ON-HANDS EXPERIENCE IN COMPUTER PROGRAMMING. WE *****
;***** CAN MAKE VERY GOOD USE OF ANY SOLS AND OTHER LIKE   *****
;***** EQUIPMENT. DONATIONS ARE TAX DEDUCTIBLE>            *****
;*****************************************************************
;*****************************************************************

;DEFINE TRUE AND FALSE.

TRUE    EQU     OFFFFH          ;DEFINE TRUE.
FALSE   EQU     NOT TRUE        ;DEFINE FALSE.

;DEFINE SOL PSEUDO PORTS.

KBD     EQU     0               ;SOL KEYBOARD.
VDM     EQU     0               ;SOL VIDEO DISPLAY.
SERIN   EQU     1               ;SOL SERIAL INPUT.
SEROUT  EQU     1               ;SOL SERIAL OUTPUT.
PARLIN  EQU     2               ;SOL PARALLEL INPUT.
```

```
PARLOUT EQU     2               ;SOL PARALLEL OUTPUT.
CSTMIN  EQU     3               ;SOL CUSTOM INPUT DRIVER.
CSTMOUT EQU     3               ;SOL CUSTOM OUTPUT DRIVER.


;*****************************************************************
;*****************************************************************
;*****  THE USER MAY WISH TO ADJUST THE FOLLOWING EQUATES  *******
;*****************************************************************
;*****************************************************************

;Change MSIZE to the desired CP/M memory size in K.

MSIZE   EQU     56              ;*** Distribution size

;SET <SHOW> TRUE IF YOU WANT TOGGLES FLAGGED IN REVERSE VIDEO.

SHOW    EQU     TRUE            ;SET FALSE ONLY TO SAVE SPACE.

;SET <SCRN16> TRUE IF 16X64, OR FALSE IF 24X80.

SCRN16  EQU     FALSE           ;*** SET TRUE IF 16X64

;SET <NULLS> TO # OF NULLS AFTER EACH LINEFEED.

NULLS   EQU     6               ;*** NUMBER OF NULLS AFTER <CR>.

;SET <DONULLS> TRUE IF YOU WANT NULLS AFTER CR.

DONULLS EQU     TRUE            ;*** SET TRUE IF NULLS ARE DESIRED.

;SET <PAGEL> TO # OF LINES TO BE PRINTED PER PAGE OF SHEET PAPER.

PAGEL   EQU     55              ;*** SET PAGE LENGTH FOR SHEET.

;SET <MAX> TO MAXIMUM # OF LINES PER PAGE (INCLUDING MARGINS).

MAX     EQU     66              ;*** SET MAX FORM LENGTH.

;ALLOW AUTOMATIC PAGE-EJECT ON RECEIPT OF LOAD/F, LOAD/S, OR LOAD/^.

NEWPAGE EQU     TRUE            ;*** SET TRUE IF NEW PAGE DESIRED.

;SET <SOLOS> TO THE BASE ADDRESS OF YOUR SOLOS ROM.

SOLOS   EQU     0F000H          ;*** BASE OF SOLOS ROM.

;SELECT WHICH PORTS/DEVICES YOU WANT FOR STANDARD CP/M I/O.

CONSIN  EQU     KBD             ;*** KEYBOARD IS USUAL CONSOLE INPUT DEVICE.
CONSOUT EQU     VDM             ;*** VIDEO IS USUAL CONSOLE OUTPUT DEVICE.
LISTOUT EQU     SEROUT          ;*** SELECT YOUR PRINTER PORT.
LISTIN  EQU     LISTOUT         ;*** NEEDED FOR XON/XOFF OR ETX/ACK
                                ;*** TYPE HANDSHAKING.
```

```
                                    ;*** <USUALLY SAME AS LISTOUT>.
RDRIN    EQU     SERIN              ;*** SELECT YOUR READER PORT.
PNCHOUT  EQU     SEROUT             ;*** SELECT YOUR PUNCH PORT.

;SET ONLY ONE OF THE FOLLOWING HANDSHAKE CODES TRUE.

NONEH    EQU     TRUE               ;*** SET TRUE IF PARALLEL OR NO HANDSHAKING.
ETXH     EQU     FALSE              ;*** SET TRUE IF USING ETX/ACK HANDSHAKING.
XONH     EQU     FALSE              ;*** SET TRUE IF USING XON/XOFF HANDSHAKING.

;SET <BUFFR> TO SIZE OF YOUR PRINTER BUFFER.

BUFFR    EQU     80                 ;*** SIZE OF PRINTER BUFFER
                                    ;***** MAXIMUM SHOULD BE 255 *****
                                    ;*** <USED BY ETX/ACK PROTOCOL>.

**********************************************************************
**********************************************************************

;THE FOLLOWING ARE FOR THOSE WHO ARE USING SUPER-PHANTOM
;OR SOME OTHER FORM OF BANK-SELECT SCHEME FOR ALLOWING
;RAM TO OVERLAY THE SOLOS ROM/RAM AREA. SET <BANK> TO <FALSE> IF
;NOT USING THIS FEATURE. IF <TRUE>, THEN FILL IN THE OTHER
;EQUATES SO BANK-SWITCHING CAN BE DONE PROPERLY.

BANK     EQU     FALSE              ;*** SET TRUE IF USING BANK-SELECT.

BNKPRT   EQU     OFFH               ;*** BANK-SELECT PORT #.
SOLON    EQU     00                 ;*** CODE TO ACTIVATE SOLOS.
SOLOFF   EQU     01                 ;*** CODE TO DE-ACTIVATE SOLOS
                                    ;***  AND ACTIVATE OVERLAY RAM.

**********************************************************************
**********************************************************************
*****     END OF EQUATES THAT MAY NEED TO BE CHANGED      *****
**********************************************************************
**********************************************************************


**********************************************************************
**********************************************************************
*****                BEGINNING OF FIXED EQUATES                 *****
**********************************************************************
**********************************************************************

;THE FOLLOWING ARE SOLOS ENTRY POINTS.

AINP     EQU     SOLOS+22H
AOUT     EQU     SOLOS+1CH
SINP     EQU     SOLOS+1FH
SOUT     EQU     SOLOS+19H

         IF      SCRN16
SYSRAM   EQU     SOLOS+800H         ;LOCATION OF SOLOS SYSTEMS RAM.
SCRNSZ   EQU     16                 ;SCREEN SIZE.
```

```
        ENDIF

        IF      NOT SCRN16      ;IF 24X80...
SYSRAM  EQU     SOLOS+0F80H     ;BOB HOGG'S MCVIDEO IS ASSUMED.
SCRNSZ  EQU     24              ;ASSUME 24X80.
        ENDIF

SOLSPD  EQU     SYSRAM+0BH      ;LOCATION OF <SPEED> BYTE.
UIPRT   EQU     SYSRAM          ;LOCATION OF CUSTOM INPUT ROUTINE.
UOPRT   EQU     SYSRAM+2        ;LOCATION OF CUSTOM OUTPUT ROUTINE.


;ASSIGN KEY/CODE VALUES.

MODE    EQU     80H             ;MODE-SELECT KEY.
CTLC    EQU     03              ;CONTROL/C.
CTLG    EQU     07H             ;CONTROL/G
DELETE  EQU     7FH             ;DELETE KEY.
LOAD    EQU     8CH             ;LOAD KEY.
LARROW  EQU     81H             ;LEFT-ARROW KEY.
FFEED   EQU     0CH             ;FORM-FEED CODE.
BS      EQU     08H             ;NORMAL BACKSPACE.
ETX     EQU     03H             ;ETX CODE.
ACK     EQU     06H             ;ACK CODE.
XON     EQU     11H             ;XON CODE.
XOFF    EQU     13H             ;XOFF CODE.
ESC     EQU     1BH             ;ESCAPE CODE.
LF      EQU     0AH             ;LINE-FEED.
CR      EQU     0DH             ;CARRIAGE-RETURN.

*****************************************************************
*****************************************************************

;The following equates are automatically changed by MSIZE.

BIOS    EQU     5300H+(MSIZE-24)*1024   ;$$$
CCP     EQU     BIOS-1600H              ;$$$
BDOS    EQU     CCP+800H                ;$$$
USER    EQU     BIOS+700H               ;$$$For double/quad density
;USER   EQU     BIOS+500H               ;$$$For single density
OFFSET  EQU     2000H-BIOS              ;$$$To overlay SYSGEN IMAGE
IOBYT   EQU     3                       ;$$$Storage location
                                        ;<<IOBYTE NOT USED >>.

*****************************************************************
*****************************************************************
*****             BEGINNING OF ACTUAL PROGRAM             *****
*****************************************************************
*****************************************************************


        ORG     USER            ;Start of USER AREA

*****************************************************************
*****************************************************************
```

```
;JUMP TABLE - Jumps MUST remain here in same order.

CINIT   JMP     CBOOT           ;COLD BOOT INIT.
WINIT   JMP     WBOOT           ;WARM BOOT INIT.
JCONST  JMP     CONST           ;CONSOLE STATUS.
JCONIN  JMP     CONIN           ;CONSOLE INPUT.
JCONOUT JMP     CONOUT          ;CONSOLE OUTPUT.
JLIST   JMP     LIST            ;LIST DEVICE.
JPUNCH  JMP     PUNCH           ;PUNCH DEVICE.
JREADER JMP     READER          ;READER DEVICE.
JLISTST JMP     LISTST          ;LIST STATUS.

;THIS DATA AREA MUST BE HERE IN THIS ORDER.

LENUA:  DW      USRLEN          ;Length of USER AREA
USRIOB: DB      80H             ;"I" Initial IOBYT
HSTYPE: DB      0FFH            ;"H" Handshaking type
NULLOC: DB      NULLS           ;"J" Printer nulls
LINES:  DB      PAGEL           ;(#lines)/page
PMAX:   DB      MAX             ;(MAX # LINES)/PAGE.

**********************************************************************
**********************************************************************

;THE FOLLOWING INITIALIZED STORAGE AREA CONTAINS ITEMS
;THAT ARE CHANGED DYNAMICALLY BY THE PROGRAM.
;*** THE USER MAY WISH TO CHANGE THE DEFAULT VALUES GIVEN.

ATOGGLE DB      00      ;*** AUTO-DISPLAY TOGGLE INITIALLY OFF.
LTOGGLE DB      01      ;*** LINE-TOGGLE INITIALLY ON.
PTOGGLE DB      00      ;*** PRINTER INITIALLY OFF.
PAGE    DB      PAGEL   ;*** PAGE SET TO FULL LENGTH.
PAPER   DB      'F'     ;*** 'F'=FANFOLD   'S'=SHEET.
LNCNT   DB      SCRNSZ-1 ;*** LINE-COUNT FOR SCREEN SCROLL.
CHRCNT  DB      BUFFR-3 ;*** CHARACTER COUNT <FOR ETX/ACK>.

**********************************************************************
**********************************************************************

;<BON> AND <BOFF> DO BANK-SELECTION OF SOLOS.

        IF      BANK            ;<ONLY NEEDED IF BANK=TRUE>.

BON:
        PUSH    PSW             ;PRESERVE "A" AND FLAGS.
        MVI     A,SOLON         ;GET CODE FOR TURNING SOLOS ROM/RAM "ON".
BSEL:
        OUT     BNKPRT          ;DO IT.
        POP     PSW             ;RECOVER "A" AND FLAGS.
        RET

BOFF:
        PUSH    PSW             ;PRESERVE "A" AND FLAGS.
        MVI     A,SOLOFF        ;GET CODE FOR TURNING SOLOS ROM/RAM "OFF".
```

```
        JMP     BSEL            ;DO IT.

        ENDIF


;*********************************************************************
;*********************************************************************

;CONSOLE STATUS ROUTINE... USES CONSIN DEVICE.

;***    <CONST> RETURNS ZFLAG ON AND "A"=00 WHEN NOT READY.
;***    IT RETURNS "A"=0FFH AND <CHARAC>=DATA WHEN READY.
;***
;***    SPECIAL CHARACTERS ARE CONVERTED/INTERPRETED AS NEEDED,
;***    AND THEN A RETURN IS MADE AS IF NOTHING HAPPENED.
;***    THIS FEATURE ALLOWS THE SPECIAL KEYS TO BE USED ANYTIME
;***    WITHOUT CAUSING ANY DIFFICULTY WITHIN CP/M.
;***
;***    SPECIAL FEATURES INCLUDE:
;***    MODE-SELECT = CONTROL/C
;***    LEFT-ARROW = BACKSPACE
;***    DELETE = DELETING BACKSPACE UNDER CP/M
;***
;***    THE LOAD KEY ACTS AS A SPECIAL "ESCAPE" KEY:
;***    LOAD/P = TOGGLE PRINTER ON/OFF
;***    LOAD/S = SHEET PAPER
;***    LOAD/F = CONTINUOUS FAN-FOLD PAPER
;***    LOAD/^ = FAN-FOLD WITH PAGE-LENGTH
;***    LOAD/A = TOGGLE SCREEN AUTO-PAGING <(HIT ANY KEY TO RESUME)>
;***    LOAD/0 THRU LOAD/9 = SET SCROLLING SPEED. 0=FASTEST
;***    LOAD/MODE = RETURN TO SOLOS
;***    LOAD/LOAD = PAUSE SCREEN
;***    LOAD/ANYTHING ELSE = PAUSE SCREEN

CONST:
        CALL    XXAINP          ;DO INPUT.
        STA     CHARAC          ;STORE RETURNED CHARACTER...
                                ;  ZFLAG IS ON AND BOTH "A" AND <CHARAC>
                                ;  ARE = 00 IF NO CHARACTER READY.
        RZ

CONSTO:
        CPI     MODE            ;CONVERT MODE-SELECT
        JNZ     CONST1
        MVI     A,CTLC          ; TO A CONTROL/C.
        JMP     GOTIT           ;WE GOT IT... RETURN TO CP/M.
CONST1:
        CPI     DELETE          ;CONVERT DELETE TO BACKSPACE.
        JNZ     CONST2
BSOUT:
        MVI     A,BS            ;(A NORMAL-TYPE BACKSPACE).
        JMP     GOTIT           ;WE GOT IT... RETURN TO CP/M.
CONST2:
        CPI     LARROW          ;ALLOW LEFT-ARROW AS BACKSPACE TOO.
        JZ      BSOUT
```

```
CONST3:
        CPI     LOAD            ;LOAD KEY IS LIKE AN "ESCAPE"...
        JZ      IN1
GOTIT:
        ANI     7FH             ;STRIP OFF MSB.
                                ;<<YOU MAY WANT TO REMOVE THE ANI 7FH>>.
        STA     CHARAC          ;SAVE IT.
LISTST:                         ;(SERVES TRIPLE DUTY).
        MVI     A,0FFH          ;TELL CP/M WE GOT A CHARACTER.
CBOOT:                          ;(CBOOT=RET).
WBOOT:                          ;(WBOOT=RET).
        RET                     ;RETURN TO CP/M.


IN1:
        CALL    XXAINP          ;<USE SOLOS>.
        JZ      IN1             ;WAIT IF NONE READY.
        ANI     5FH             ;STRIP IT.
                                ;<<CONVERT LOWER TO UPPER CASE>>.
        CPI     '0' AND 5FH     ;LOAD/0 = FASTEST SPEED.
                                ;LOAD/9 = SLOWEST SPEED.
        JC      IN2             ;IF LESS THAN '0', NOT A SPEED.
        CPI     ('9'+1) AND 5FH
        JNC     IN2             ;IF >9 THEN NOT A SPEED.
        ANI     0FH             ;ALL WE NEED IS LAST PART.
        IF      BANK            ;<ONLY NEEDED IF BANK=TRUE>.
        CALL    BON             ;TURN SOLOS ROM/RAM ON.
        ENDIF
        STA     SOLSPD          ;STUFF IT INTO SPEED REGISTER.
        IF      BANK            ;<ONLY NEEDED IF BANK=TRUE>.
        CALL    BOFF            ;TURN SOLOS ROM/RAM OFF.
        ENDIF
        JMP     NOTHING         ;ACT LIKE NOTHING HAPPENED.
IN2:
        CPI     'S' AND 5FH     ;LOAD/S = SET FOR SINGLE-SHEET.
        JNZ     IN3
SETUP:
        PUSH    PSW             ;SAVE CODE.
        MVI     A,1             ;<LOAD/S AND LOAD/^ SET LTOGGLE ON>.
SETUP1:
        STA     LTOGGLE
        POP     PSW             ;RECOVER CODE.
        STA     PAPER           ;SET PAPER TYPE.
        IF      NOT NEWPAGE     ;<CONDITIONAL CLEAN-UP>
        LDA     LINES           ;SET UP FRESH PAGE.
        STA     PAGE
        ENDIF
        IF      NEWPAGE         ;<CONDITIONAL PAPER-EJECT>.
        PUSH    B
        MVI     C,FFEED
        CALL    LIST            ;FORCE A PAGE EJECT ON LIST DEVICE.
        POP     B
        ENDIF
        JMP     NOTHING         ;ACT LIKE NOTHING HAPPENED.
IN3:
```

```
        CPI     '^' AND 5FH     ;LOAD/^ = SET FOR FAN-FOLD/P.
        JZ      SETUP           ;FINISH ELSEWHERE.
IN4:
        CPI     MODE AND 5FH    ;LOAD/MODE-SELECT = GO TO SOLOS.
        JNZ     IN5
        IF      BANK            ;TURN ON SOLOS FIRST.
        CALL    BON
        ENDIF
        JMP     SOLOS+4         ;USE PROPER RETURN POINT.
IN5:
        CPI     'A' AND 5FH     ;LOAD/A = TOGGLE AUTO-PAUSE.
        JNZ     IN6
        MVI     A,SCRNSZ-1      ;RESET LINE-COUNT.
        STA     LNCNT
        LDA     ATOGGLE         ;GET CURRENT STATUS.
        INR     A               ;UPDATE IT.
        STA     ATOGGLE         ;STORE IT.
        JMP     PNOTHING        ;ACT LIKE NOTHING HAPPENED.
IN6:
        CPI     'F' AND 5FH     ;LOAD/F = CONTINUOUS FAN-FOLD PAPER.
        JNZ     IN7
        PUSH    PSW             ;SAVE CODE.
        XRA     A               ;RESET LTOGGLE.
        JMP     SETUP1          ;FINISH UP ELSEWHERE.
IN7:
        CPI     'P' AND 5FH     ;LOAD/P = TOGGLE PRINTER.
        JNZ     NOTHING         ;LOAD/ANYTHING ELSE = PAUSE.
        LDA     PTOGGLE         ;GET CURRENT P-TOGGLE STATUS
        INR     A               ; TOGGLE IT.
        STA     PTOGGLE         ;MAKE THAT PERMANENT.
PNOTHING:
        IF      SHOW

        PUSH    B
        PUSH    PSW             ;SAVE TOGGLE VALUE.
        MVI     C,ESC           ;SEND ESCAPE CODE TO SOLOS.
        XRA     A
        CALL    XAOUT
        MVI     C,CTLG          ;SEND CONTROL/G TO SOLOS
        XRA     A
        CALL    XAOUT           ;<ALLOWS REVERSE-VIDEO>.
        POP     PSW             ;RECOVER TOGGLE VALUE.
        ANI     01              ;LOOK AT LSB ONLY.
        ADI     '0' OR 80H      ;SEND REVERSE-VIDEO "0" OR "1".
        MOV     C,A             ;PRINT IT FROM C.
        XRA     A
        CALL    XAOUT
        POP     B

        ENDIF


NOTHING:
        XRA     A               ;SET ZFLAG FOR SURE.
SCHARAC:
```

```
        ORA     A               ;SET ZFLAG IF A=00
        PUSH    PSW             ;(SEVERAL ROUTINES USE <SCHARAC>).
        XRA     A
        STA     CHARAC          ; RESET <CHARAC>.
        POP     PSW
        RET                     ;ZFLAG IS ON AND "A"=00.
                                ;(IF ENTERED THROUGH <NOTHING>).

**********************************************************************
**********************************************************************

;CONSOLE INPUT- ROUTINE.

*** THE CONSOLE INPUT ROUTINE RELIES HEAVILY UPON THE
*** <CONST> ROUTINE. ALL CODE CONVERSIONS AND CHECKS FOR
*** SPECIAL CHARACTERS ARE DONE INSIDE <CONST>.

CONIN:
        LDA     CHARAC          ;CHECK FOR A CHARACTER FROM CONST.
        CALL    SCHARAC         ;RESET <CHARAC>.
                                ;(AND SET Z FLAG IF NO CHARACTER).
        RNZ                     ;IF WE GOT SOMETHING, LOOK NO FURTHER.
        CALL    CONST           ;ULTIMATELY IT COMES FROM CONST.
        JMP     CONIN

**********************************************************************
**********************************************************************

;CONSOLE OUTPUT ROUTINE.

*** BACKSPACES ARE PROPERLY HANDLED.
*** MULTIPLE CARRIAGE-RETURNS ARE IGNORED.
*** SEE CONST FOR DETAILS ON PAUSING DISPLAY AND SETTING SPEED.
***

CONOUT:
        LDA     PTOGGLE         ;CHECK PRINTER TOGGLE.
        ANI     1               ;ONLY LSB IS USED.
        CNZ     LIST            ;IF ON, LIST IT.
OUT1:
        PUSH    B               ;SAVE ORIGINAL.
        MOV     A,C             ;WE NEED IT IN "A" FOR COMPARES.
        CPI     BS              ;SOL NEEDS HELP WITH BACKSPACE.
        JNZ     OUT2
        MVI     C,LARROW        ;CONVERT TO SOL LEFT-ARROW.
OUT2:
        CPI     CR              ;DON'T ALLOW CR/CR.
        JNZ     SENDIT
        LDA     LCHAR           ;RECOVER LAST CHARACTER DISPLAYED.
        CPI     CR
        JNZ     SENDIT          ;TRAP CR/CR
        JMP     POPB            ;JUST SKIP DISPLAYING THEM.
SENDIT:
        STA     LCHAR           ;SAVE LAST CHARACTER DISPLAYED.
        MVI     A,CONSOUT       ;OUTPUT TO SELECTED DEVICE.
```

```
        CALL    XAOUT
        CPI     LF              ;LINE-FEEDS ARE SPECIAL.
        JNZ     POPB
        LDA     ATOGGLE         ;GET AUTO-DISPLAY TOGGLE.
        ANI     1               ;JUST LSB.
        JZ      POPB            ;IGNORE IF OFF.
        LDA     LNCNT
        DCR     A               ;UPDATE LINECOUNT.
        STA     LNCNT
        JNZ     POPB            ;NO ACTION UNTIL = 0.
        MVI     A,SCRNSZ-1      ;RESET LINECOUNT TO SCREENSIZE.
        STA     LNCNT
        CALL    CONIN           ;...PAUSE UNTIL A KEY IS HIT.
                                ;<ALSO ALLOWS SPECIAL KEYS>.
POPB:
        POP     B               ;RECOVER ORIGINAL.
RSTORE:
        MOV     A,C             ;CP/M LIKES IT IN "A".
        RET

;*********************************************************************
;*********************************************************************

;LIST ROUTINE.

LIST:
        PUSH    B               ;SAVE BC.
FFCHK:
        MOV     A,C             ;CHECK FOR FORMFEED.
        CPI     FFEED
        JNZ     LISTIT          ;IF NOT FORMFEED, CONTINUE.
FFCHK1:                         ;IF FORMFEED, HANDLE SPECIAL.
        LDA     LINES           ;GET NORMAL PRINT LENGTH...
        CMA                     ;MAKE IT NEGATIVE...
        INR     A               ;...TWO'S COMPLEMENT TYPE...
        MOV     B,A
        LDA     PMAX            ;GET <MAX> FORM LENGTH...
        ADD     B               ;CALCULATE MAX-LINES...
        MOV     B,A
        LDA     PAGE            ;GET LINES LEFT TO PRINT...
        ADD     B               ;CALCULATE LINES FOR FORMFEED.
        INR     A               ;ALLOW A CR TOO.
        PUSH    PSW             ;SAVE IT.
        MVI     A,0FFH
        STA     PAGE            ;PREVENT "HICCUPS".
        POP     PSW             ;RECOVER FORMFEED LENGTH.
        MVI     C,CR            ;ALWAYS DO ONE CARRIAGE-RETURN.
FFLOOP:
        PUSH    PSW             ;SAVE COUNT.
        CALL    LIST            ;PRINT A LINEFEED (OR CR).
        MVI     C,LF            ;NEXT TIME IT'S A LINE-FEED.
        POP     PSW             ;RECOVER COUNT.
        DCR     A               ;UPDATE IT.
        JNZ     FFLOOP          ;DO LINE-FEEDS AS NEEDED.
LWAIT:
```

```
        LDA     LINES           ;RESET LINE COUNTER.
        STA     PAGE
        LDA     PAPER           ;CHECK WHAT KIND OF PAPER...
        CPI     'S' AND 5FH     ;...IS IT SHEET?
        JNZ     POPB            ;IF NOT, ALL DONE.
        CALL    CONIN           ;IF SHEET, WAIT FOR ANY KEY TO BE HIT...
                                ;...<ALLOW SPECIAL FUNCTIONS TOO>.
        JMP     POPB            ;WHEN DONE, BE NEAT.


LISTIT:
        IF      XONH            ;XON/XOFF IS DONE EARLY.

XOFFL:
        MVI     A,LISTIN        ;IS XOFF BEING RECEIVED?
        CALL    XAINP
        JZ      CNTNU           ;IF NOT, GO AHEAD.
        ANI     7FH             ;STRIP OFF MSB.
        CPI     XOFF            ;MAKE SURE IT'S XOFF.
        JNZ     CNTNU           ;IGNORE IF NOT.
XONL:
        MVI     A,LISTIN        ;IS XON BEING RECEIVED?
        CALL    XAINP
        JZ      XONL            ;IF NOT, WAIT.
        ANI     7FH             ;STRIP OFF MSB.
        CPI     XON
        JNZ     XONL            ;WAIT UNTIL XON.
CNTNU:                          ;OK, GO ON.

        ENDIF                   ;END OF XONH ROUTINE.

        CALL    XLAOUT          ;USE SELECTED LIST DEVICE.

        IF      ETXH            ;USE ETX/ACK PROTOCOL.
                                ;THIS VERSION INCLUDES CHECK FOR ESC
                                ;SO ESCAPE CODES GET THROUGH OK.

        MOV     A,C             ;GET CHARACTER INTO "A".
        CPI     ESC             ;ESCAPE NEEDS HELP.
        JNZ     ETX1
        LDA     CHRCNT          ;GET CURRENT COUNT.
        ADI     3               ;ADD 3 TO IT...
        STA     CHRCNT          ;... SO ESC SEQUENCE MAKES IT.
ETX1:
        LDA     CHRCNT          ;GET CHARACTER COUNT.
        DCR     A               ;UPDATE IT.
        STA     CHRCNT          ;SAVE IT.
        JNZ     LISTON          ;IF NOT EMPTY, GO HOME MAYBE.
        MVI     A,BUFFR-3       ;RESET CHRCNT.
        STA     CHRCNT
ETXL:
        MVI     C,ETX           ;SEND ETX CODE.
        CALL    XLAOUT          ;USE SELECTED LIST DEVICE.
ACKL:
        MVI     A,LISTIN        ;GET INPUT FROM LIST DEVICE.
```

```
        CALL    XAINP
        JZ      ACKL            ;WAIT IF NECESSARY.
        ANI     7FH             ;STRIP OFF MSB.
        CPI     ACK             ;IF NOT ACK...
        JNZ     ACKL            ;... WAIT 'TIL IT IS.

        ENDIF                   ;END OF ETXH ROUTINE.


LISTON:
        POP     B               ;RECOVER ORIGINAL CHARACTER.
        PUSH    B               ;PUT IT BACK FOR NOW.

        IF      DONULLS         ;<CONDITIONAL> NULL GENERATOR.
        MVI     A,CR            ;DO NULLS AFTER <CR>.
        CMP     C
        JNZ     NONULLS
;***** SPECIAL NULL ROUTINE *****
        LDA     NULLOC          ;GET NULL COUNT.
        MOV     B,A             ;<B> IS HARMLESS.
        PUSH    B               ;DON'T LOSE IT.
NULLOOP:
        MVI     C,0             ;SEND A NULL.
        CALL    XLAOUT          ;USE SELECTED LIST DEVICE.
        POP     B               ;RECOVER THE COUNTER.
        DCR     B               ;UPDATE COUNT.
        PUSH    B               ;SAVE IT.
        JNZ     NULLOOP         ;DO SEVERAL.
        POP     B               ;KEEP THE STACK CLEAN.
;***** END OF NULL ROUTINE *****
        ENDIF

NONULLS:
        LDA     LTOGGLE
        ORA     A
        JZ      POPB
LFEED:
        MOV     A,C             ;USE IT FOR COMPARES.
        CPI     LF              ;LINE-FEEDS ARE ALSO SPECIAL.
        JNZ     POPB
        LDA     PAGE            ;GET PAGE LENGTH LEFT.
        DCR     A               ;REDUCE BY ONE.
        STA     PAGE            ;STORE IT.
        JNZ     POPB            ;IF DONE, BE NEAT.
        JMP     FFCHK1          ;DO A FORMFEED.

;*****************************************************************
;*****************************************************************

;PUNCH IS USEFUL FOR DRIVING MODEMS.

PUNCH:
        MVI     A,PNCHOUT       ;PUNCH IS WHATEVER YOU SELECTED.
        JMP     XAOUT
```

```
;*********************************************************************
;*********************************************************************

;READER IS USEFUL FOR LISTENING TO MODEMS.

READER:
        MVI     A,RDRIN         ;READER IS WHATEVER YOU SELECTED.
        CALL    XAINP
        JZ      READER          ;WAIT UNTIL YOU GET SOMETHING.
        RET

;*********************************************************************
;*********************************************************************
;*****   THE FOLLOWING ROUTINES COMMUNICATE WITH SOLOS.    *****
;*****   THEY ALLOW BANK-SELECT <SUCH AS SUPER-PHANTOM>    *****
;*********************************************************************
;*********************************************************************

;XAOUT ALLOWS ANY SOL PSEUDO OUTPUT PORT TO BE USED FOR OUTPUT.

*** ENTER WITH PSEUDO OUTPUT PORT # IN "A".
*** 0 = VDM
*** 1 = SERIAL PORT
*** 2 = PARALLEL PORT
*** 3 = CUSTOM DRIVER <<YOU MUST DEFINE CUSTOUT=CUSTOM ADDRESS>>

XLAOUT:
        MVI     A,LISTOUT
XAOUT:
        IF      BANK            ;<ONLY NEEDED IF BANK=TRUE>.
        CALL    BON             ;TURN SOLOS ROM/RAM ON.
        ENDIF
        PUSH    B
        MOV     B,C             ;SOLOS NEEDS IT IN "B".
        CALL    AOUT            ;USE SPECIFIED PSEUDO PORT.
        POP     B
        IF      BANK            ;<ONLY NEEDED IF BANK=TRUE>.
        CALL    BOFF            ;TURN SOLOS ROM/RAM OFF.
        ENDIF
        MOV     A,C             ;CP/M LIKES IT IN "A".
        RET

;*********************************************************************
;*********************************************************************

;XAINP ALLOWS ANY SOL PSEUDO INPUT PORT TO BE USED FOR INPUT.

*** ENTER WITH PSEUDO OUTPUT PORT # IN "A".
*** 0 = KEYBOARD
*** 1 = SERIAL PORT
*** 2 = PARALLEL PORT
*** 3 = CUSTOM DRIVER <<YOU MUST DEFINE CUSTIN=CUSTOM ADDRESS>>.

XXAINP:
        MVI     A,CONSIN        ;PREPARE FOR CONSOLE INPUT.
```

```
XAINP:
        IF      BANK            ;<ONLY NEEDED IF BANK=TRUE>.
        CALL    BON             ;TURN SOLOS ROM/RAM ON.
        ENDIF
XAINP1:
        CALL    AINP            ;USE SOLOS AINP PSEUDO PORT.
        IF      BANK            ;<ONLY NEEDED IF BANK=TRUE>.
        CALL    BOFF            ;TURN SOLOS ROM/RAM OFF.
        ENDIF
        RET

*************************************************************************
*************************************************************************
*************************************************************************
*************************************************************************
*****                      END OF PROGRAM                           *****
*************************************************************************
*************************************************************************
*************************************************************************
*************************************************************************

;THE FOLLOWING ARE UNITIALIZED DATA.

CHARAC  DB      00      ;*** CHARACTER FROM CONST.
LCHAR   DB      00      ;*** LAST CHARACTER DISPLAYED.

*** THE FOLLOWING DETERMINE THE AMOUNT OF USER AREA LEFT.

ENDUSER EQU     $
MAXLEN  EQU     512
USRLEN  EQU     ENDUSER-USER
LFTOVR  EQU     MAXLEN+2-USRLEN
;IF <LFTOVR>=FXXX THEN OPTIONS SELECTED HAVE CAUSED
;LENGTH TO EXCEED 512 BYTES. TRY CHANGING OPTIONS.


*************************************************************************
*************************************************************************

        END

C>
```

April 5, 1984

Stanley M. Sokolow, D.D.S.
PROTEUS
1690 Woodside Road, Suite 219
Redwood City, CA 94061

Dear Stan,

The accompanying diskette contains Franz J. Hirner's General Ledger system programs in Processor Technology's EDBASIC source. I obtained printed listings from Mr. Hirner in April, 1983, and have been entering and testing the programs since then. You may copy the diskette for Proteus members as you see fit. If you need a formal release from Mr. Hirner, please contact him.

File 'message2' on the diskette contains information about the system. Enter 'BO' to see it during START.UP or print it directly on screen or on hardcopy. The original reference to the system appeared in Proteus September/October/November/ December 1981, Vol. 4, #5/6, pp. 3-5. It is a very nice addition to my collection of application programs. I hope other Proteus members like the system as much as I do.

Sincerely,

Leonard

Leonard E. Cole
644 San Fernando Ave.
Berkeley, CA 94707
Phone: 415-527-2110

cc: Mr. Franz J. Hirner
631 Matsonia Drive
Foster City, CA 94404

Editor's note ...

Please contact Franz Horner or Leonard Cole to obtain this disc.

WILLIAM D. LOUGHMAN
GENETICIST
393 GRAVATT DRIVE · BERKELEY, CALIFORNIA 94705 · (415) 841-7428

20 July 1984

Dr. Stan Sokolow
PROTEUS, Inc.
1690 Woodside Road, Suite 219
Redwood City, CA 94061

Dear Stan:

As you have said, Proteus may 'wind down'. Given your own steady investment in time and energy, to say nothing of the head-over-heels advancement of the entire micro-computer industry, that can't be too surprising. But there remains a place for Proteus, and I cast my vote for continuation. Perhaps as a quarterly, for those of us who value it enough to pay (and contribute). Perhaps expanded, as a newsletter for other 'antique' machines. There are only two publications that have been of outstanding value to me: Proteus News, and Dr. Dobbs (I subscribe to over a dozen more).

So in my view, Proteus should continue. For one, I'll pay increased fees if necessary. The SOL remains a practical machine, and like you, Stan, I intend to keep my investment useful. It continues to be a capable and very adaptable machine.

As to contributing ... Well, yes, this letter is years over-due. We've talked on the phone many times over the years, but I've written only one letter (a CP/M BIOS modification). That generated several fast responses, so I felt pleased to have helped others, and guilty that I hadn't done more. In some measure, this letter is meant to assuage my guilt. In some measure, I hope its tone will encourage others to write.

To add some substance to my own desire for more 'collegiality', let me detail a few of the areas in which I might help (local) SOL owners.

I own the Buss Probe (an auxiliary board from Jade which displays status of and signals on the S-100 buss), the ParaSol board and software (a PTCo machine debugger), a logic probe, and a fast triggered 'scope. My acquaintance with computer hardware is distant, but these, the Sol manual, and some thoughtful jiggling have helped me more than once to avoid expensive down-time.

My special delight, computer-wise, runs to assembler code. I've written BIOSs, full-screen editors, assemblers, disassemblers, telecommunication modules, and even a complete operating system. Time permitting, I could help those who have some problems with interfacing

the SOL to commercial software.

As to common commercial (or other) software, there are Things I Like, and Things I Don't Like.

Things I Like (and use a lot):
Spellbinder (word-processor - see below); Supercalc (spreadsheet); Modem7 (telecommunication); and the SOLOS monitor (MicroComplex mod). In re Spellbinder: Their customer service is the pits. There is a bright spot on the Lexisoft stonewall however: The current Female Voice On The Phone. Don't try to bypass this gem, ask HER to solve your problems. She's good, and deserves a raise.

Things I Like (and use a little):
PILOT (Starkweather's 'Nevada' beta-test version, also Stok PILOT); NorthStar Basic (SoHo CP/M conversion); PTCo BASIC (for the matrix functions); and fig-FORTH.

Things I Like (I think... but only a little experience with):
BDS 'C' (not quite Kernighan and Ritchie, but a huge base of programs; and it is fast).

Utilities that are 'the greatest':
DUxx (CP/M disk editor - Ward's second greatest gift); and CATxx (disk catalog - Wards third greatest gift). Maybe best of all: like Modemxx, these useful and reliable programs are 'public domain'.

Which brings me to People I Like:
First, Stan Sokolow (alias SOLUS, alias PROTEUS). This newsletter and its past and ongoing contents justify that 'First'. Ward Christiansen, for reasons cited above. Bob Hogg (MicroComplex), for continuing to support the SOL. I have some of his upgrades; they work reliably and as advertised, and make a hell of a difference.

Finally, Things I Don't Like: (and avoid whenever possible):
The absolute FIRST dislike has to be dBaseII v.2.3B and v.2.4. Ashton-Tate is a marketing corporation, not a software author. The difference shows in their regard for both their product and their customers. I haven't the space here to say much more, but the product is UNRELIABLE. It trashes disk files in unpredictable ways, at unpredictable times. I've circulated a letter to PICONET and the San Francisco dBase User's Group detailing my experiences, and some experiences of others, and my conversations with Ashton-Tate. There is a popular book on dBase which supports my views, and so does an article in the current 'Computer Currents' (a Bay Area tabloid). Further, A-T's Chief Programmer called me on 22 June 84, confirming my views. Should any readers wish more information, I'll try to comply. SEND ENOUGH POSTAGE, with a SASE, to cover a 9-page document plus miscellaneous supporting material.

WordStar is another, though lesser, marketing triumph. Too big, somewhat clumsy, and 'busy-busy'. Wordmaster, Nevada EDIT, or even

Electric Pencil are better (faster) editors; Spellbinder is a better and faster word-processor. Spellbinder topped WordStar in a poll of 'computer-professionals' published last year in InfoWorld.

My final dislike is aimed at all those programs published by people who simply want their names in print. In my experience, at least half simply don't work. The rest often require extensive debugging. There is a rare well-written gem, and I am thankful for same. But they ARE rare. (My own contribution to PROTEUS has an error in the documentation - my fault - but the program works as written.)

All of the programs cited above, and dozens more, I've put onto my machine. The point here is NOT to toot my own horn, but rather to offer my experience to others. Several people have written PROTEUS, concerned that they couldn't get SuperCalc working. I have a CP/M BIOS (NorthStar) which allows one to run SuperCalc or dBase on a SOL. Ask. PROTEUS/NEWS is a newsletter, after all.

So I promised to say more about the SOL. Indirectly, I just did. Perhaps the bottom line has something to do with the following:

I use two 32k Godbout boards with the MicroComplex VDM mods to get a 64k SOL, and an aging NorthStar disk system for over a half-megabyte external storage. Thus I have two slots and the connector atop the backplane for expansion (bank select, color, etc.; whatever I want). The serial and parallel ports don't have to be added, and I have cassette I/O too (for archive at 20% of disk costs). There are LOTS of newer machines that can't offer that, even if you pay extra. Sure, the SOL is aging... but like a fine wine. We all should age so well. If you don't believe me, ask Bob Hogg (McSOLOS, McVideo, etc.) who has a wealth of machines to play with. His preference for real work remains... yes, Virginia... the SOL. My own friends often ask me to find/fix some problem for their 'modern' machines. I just don't have the heart to tell them the solution came out of a seven-year old 'antique'.

Sincerely,

Bill

William D. Loughman, Ph.D.

Burglary Notice

Would you let this man, Glen T. Buie, work on your Sol?

Just thought you would like to know where he got all of the spare parts from. I hope you let the Public know about this burglary.

(Anonymous)

Thank You.

Stolen computer gear found; man jailed

ARLINGTON — An Arlington man was arrested on suspicion of burglary Wednesday and $74,000 in computer equipment was recovered from his apartment, police said.

Deputy Police Chief Marion Rettig said investigators will consult with the district attorney's office Thursday on possible charges to be filed.

"It's most unusual for us to have computer equipment stolen, but I suspect we'll see it more and more now that people are getting into the home computer business," Rettig said.

Burglars went through the roof of Computer Port, 2142 N. Collins, during the weekend, and the burglary was discovered Monday, police said.

Taken in the theft were visual monitors, Apple II computers and Horizon computer units, police report.

Computers recovered; man arrested

ARLINGTON — Police here have recovered $74,000 in computer equipment in an apartment of an Arlington man arrested on suspicion of burglary.

Deputy Police Chief Marion Rettig said investigators were expected to consult today with the district attorney's office as to what charges might be filed.

Someone went through the roof of Computer Port, 2142 N. Collins, during the weekend, and the burglary was discovered Monday, police said.

Taken in the theft were visual monitors, Apple II computers and Horizon computer units, police report. The items were recovered in the attic of the suspect's apartment, Rettig said.

Burglar's defense alarming excuse

By JIM MORRIS
Star-Telegram Writer

An Arlington computer technician found guilty Wednesday of stealing $75,000 in computer equipment from his former employer claimed he was simply trying to convince his former boss to purchase a burglar alarm system.

The testimony of defendant Glen T. Buie failed to convince Judge Gordon Gray Wednesday in state district court, however.

After hearing testimony in Buie's burglary trial, Gray pronounced the defendant "insane."

"I'd send him to the pen in a New York minute if he didn't have the capability of making a decent citizen," Gray ruled.

Gray assessed a six-year probated sentence and a $500 fine and ordered Buie to make $3,000 restitution. In addition, the judge said, Buie must spend 30 days in jail as a term of his probation.

Buie, 25, of 315-B N. Davis, was charged with burglarizing Computer Port in Arlington on Aug. 8. Buie admitted Wednesday that he had been fired immediately prior to the burglary. However, he said he decided to take the equipment because he had been unable to convince the store's owner to install a burglar alarm system that Buie had been designing.

Buie denied he was seeking revenge on his employer and said he did not intend to keep the various computers, terminals, disc drives and monitors he swiped. The equipment was returned to the store after police discovered it in Buie's apartment.

Buie said another man, Charles Rawls, entered the Computer Port building through an air conditioning duct Aug. 8 and then let him in through a side door. The two spent the next three hours loading the delicate equipment into their cars, Buie said, before taking it to his apartment.

When police searched the apartment Aug. 11, they found the equipment mostly undamaged in the attic.

"I told (store owner Phil) Dorcas he was inviting trouble without some sort of security system," Buie testified Wednesday. "I did it to prove a point to the owner . . . that he needed a burglar alarm system."

Defense attorney Joe Shannon asked Buie why he took so many valuable items.

"I wanted to show them that a burglary of that magnitude could wipe them out," Buie said.

After Shannon had rested his case, Gray said he thought Buie was "definitely insane to do that. I think he was fixing to go into business for himself. It was a novel, ingenious defense."

Dorcas earlier had testified that had the equipment not been returned, Computer Port would have gone bankrupt. Dorcas said he had agreed with Buie that the store needed a security system. Dorcas said he had permitted Buie to continue working on the system even after Buie had been fired.

The store owner said only the most current, salable items were taken. The older, "obsolete" equipment was untouched, Dorcas said.

Shannon noted that Buie voluntarily allowed police to search his apartment and made sure all the stolen items were returned to the store.

Assistant District Attorney Paul Conner asked why Buie initially had placed the blame on Rawls, who pleaded guilty to burglary last month and received a four-year probated sentence and was ordered to make $1,000 restitution. Buie had told police they should question Rawls and friends of his who were members of a "witchcraft cult."

Buie said he wanted to "buy us some time" until he could talk to Dorcas about returning the equipment.

In his closing argument, Conner contended Buie stole the merchandise to start his own computer repair business. Computer Port manager Jay Weiner had testified that there is a "good market" for second-hand computers.

Shannon told Gray that Buie was guilty of using "darned poor judgment," not burglary. "He's obviously not a criminal because the planning was terrible," Shannon said.

Conner, however, said Buie concocted his story to "save his hide."

Buie declined to comment after being sentenced. On the witness stand, he said, "I went about this in a totally dumb, stupid manner. If I had it to do over again, I definitely would think of some other way."

SHEET NR: 2 OF 9

R/J#

EVIDENCE TRANSMITTAL SHEET

ARLINGTON
POLICE DEPARTMENT

CALL# 810810049

RELATED OFFENSE NR:

1-TYPE OF OFFENSE Burglary Business DATE 8 / 10/81

2-LOCATION OF OCCURRENCE 2142 N. Collins

3-WEATHER CONDITION AT TIME OF OFFENSE: Hot and Dry

4-ARRESTEE: Glen T. Buie and Charles E. Rawls

5-VICTIM: Computer Port RES.PHONE:

CODES: P=PROPERTY ROOM L=LABORATORY R=RELEASED TO OWNER OR REPRESENTATIVE

ITEM	DESCRIPTION	CODE	PROP.USE ONLY
08 ✓	Disk Drive Model 277 S/N 1465	P	
09 ✓	Disk Drive Model 270 S/N 2382	P	
10 ✓	Disk Drive Model 270 S/N 4526	P	
11 ✓	Disk Drive Model 270 S/N 4541	P	
12 ✓	Disk Drive Model 270 S/N 4381	P	
13 ✓	Disk Drive Model 270 S/N 4680	P	
14 ✓	Disk Drive Model 270 S/N 2693	P	
15 ✓	Helios II Disk Drive Memory System S/N 501512	P	
16 ✓	Helios II Disk Drive Memory System S/N 501484	P	
17 ✓	Helios II Disk Drive Memory System S/N 501763	P	
18 ✓	Sol Terminal Computer Model 20 S/N 401139	P	

6-REMARKS:

7-CHAIN OF CUSTODY

ITEMS (8-18): RELEASED TO: Property Room DATE: 8 / 13 / 81 TIME: 1400 AM/PM

PURPOSE: Evidence Storage

ITEMS (8-18): RELEASED TO: DATE 8/15/81 TIME: 1000 AM/PM

PURPOSE: return to owner

ITEMS ( ): RELEASED TO: DATE / / TIME: AM/PM

PURPOSE:

ITEMS ( ): RELEASED TO: DATE / / TIME: AM/PM

PURPOSE:

ITEMS ( ): RELEASED TO: DATE / / TIME: AM/PM

PURPOSE:

8- ITEMS ENTERED INTO EVIDENCE BY: D.M. Sustaire ID# 147

DATE: 8 / 13 / 81 TIME: 1400 AM/PM

Announcements

1) COMPUTER LANGUAGE is a new magazine especially for people who write serious code. It covers major development in the software design field. For more information, contact Computer Language, 2443 Fillmore St, 346 San Francisco, CA 94115.

2) World Disk Drives has purchased the manufacturing rights to all Siemens 8" disk drives, including the FDD100-8 and the FDD200-8. Special prices are available to clubs.

SIEMENS FDD200-8 (single or double density)

1-50 -------- $169.00

51-100-------- $159.00

100+ -------- $$SPECIAL DISCOUNT DEPENDING ON SIZE

SIEMENS FDD100-8 (double density)

1-50 -------- $154.00

51-100-------- $144.00

100+ -------- $$SPECIAL DISCOUNT DEPENDING ON SIZE

ALSO! Included in this special offer: (1 of each with each drive)

*1---COUPON FOR 30% OFF ON PARTS

*2---60 DAY GUARANTEE

*3---COUPON FOR 1 DRIVE REPAIR FOR $80.00

*4---1 FREE OPERATORS MANUAL (ORIGINAL)

*5---ORIGINAL MANUFACTURERS MAINTENANCE MANUAL

(INCLUDES SCHEMATIC) FOR ONLY $15.00

For more information contact World Disk Drives, P.O. Box 2000-154, Mission Viejo, CA 92690; (714) 855-1761.

3) Electronic Specialists, Inc., now offer a KLEEN LINE MODEM protection. Models are available for standard 4 pin telephone modular connectors (RJ-11) and wider professional 8 pin connectors (RJ-45). These are intended to suppress software altering telephone and power line spikes caused by lightning, spherics, or phone office switch gear. For more information, contact Frank Stifter, Electronic Specialists, Inc., 171 South Main Street, Natick, Mass. 01760, (617)655-1532.

4) This will be the last issue for Volume 6 of PROTEUS NEWS. PROTEUS members who have paid dues for 1984 (amount same as 1983) will continue to receive flyers and announcements at various times throughout the year, however a formal newsletter will not be assembled unless more contributions from members are received.

TABLE OF CONTENTS

A Farewell Note by Stan Sokolow (Editor) 1
McZol: The Z-80 Upgrade For SOL From MICROCOMPLEX 3
Unclassified Ad 5
Super-User Area for Lifeboat CP/M by Fr. McGahee 6
Hirner's General Ledger System by Leonard Cole 28
SOL Is A Valuable Antique by Wm. Loughman 29
Burglary Notice 32
Announcements 35

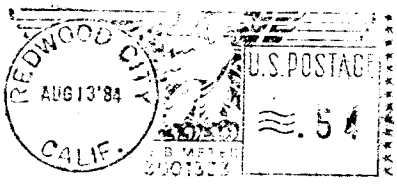
P R O T E U S / N E W S

A news journal for owners and users of Processor Technology Corporation computer equipment. Published by Proteus, 1690 Woodside Road, Suite 219, Redwood City, California 94061-3483,USA, telephone (415) 368-2300.

Submit items for publication to Proteus News, Attn: Stan Sokolow, 1690 Woodside Road, Suite 219, Redwood City, California 94061-3483, USA. Please make submissions as camera-ready as possible by using a fresh, black ribbon and typing single-spaced.

FROM:
PROTEUS
1690 WOODSIDE ROAD, SUITE 219
REDWOOD CITY, CALIFORNIA 94061-3483
USA

FIRST CLASS MAIL



James D. McElroy
2826 Crest Ave. North
Allentown, PA

18104