July/Aug 1989

USUS NewsLetter

Copyright 1989 USUS, Inc.

All the News that Fits, We Print

William D. Smith, Editor

Volume 3
Number 6

From the Editor

by William D. Smith

I received two offers of help in checking out the manuals donated to USUS by Dr. Thibaud. One from **Harry W. McMackin, III** and one from **Bernard Perrenoud**. Thanks to both. Since I received Harry's letter first, I sent him a copy of the "Table of Contents". His response was that the manuals "...appeared to be a reference book for first-year college students. It is unlike a textbook in that it does not have problems to wok through." It is also specific for the Rainbow-100 system. If there is anyone out there who has a use for these manuals, let me know.

The next issue of the NewsLetter (Sept/Oct 89) will be the election issue. There are two positions on the Board of Directors up for grabs. USUS needs new people to bring new life into the organization. Preferably those intrested in structured languages, but not necessarily in the p-System. If you are interested or know someone who is, please contact **Alex Kleider** at (415) 327-7916.

After doing this NewsLetter for a year and producing it both every month (November thru March) and every other month (May/Jun, Jul/Aug, Sep/Oct), I have decided that producing a NewsLetter every month takes up too much of my time, therefore next year there will only be six issues of the NewsLetter (Jan/Feb, Mar/Apr, May/Jun, Jul/Aug, Sep/Oct and Nov/Dec).

Should USUS have a journal?

by A. Robert Spitzer

USUS membership and renewal rates have been declining in recent years. One of the important questions that comes up is, "what does USUS do for its membership?" Obviously, many folks feel the answer is "not enough", and are 'voting with their feet'. Several proposals have been made in this regard.

One such proposal is that we develop a formal, official relationship with the Journal of Pascal, Ada and Modula-2 (JPAM). The theory is that most professional societies really only provide 2 major services to their membership. The first is their publications, generally a professional journal dealing with the membership's major interests, and often additional brief newsletters; and the second is annual meetings.

A proposal has been made to make JPAM the official journal of USUS. This would really primarily mean that the journal would be bundled in with the membership. Based on preliminary discussions, there are several approaches that would be feasible. The specific proposal that I made, is that all members receive the journal with their membership. The journal now costs $26/year. We can get it for $20/year. If we raised the annual dues from $35 to $45 a year, but included the subscription in that price, this would leave $25 per member for USUS to use.

Obviously, the immediate effect is to reduce USUS income somewhat, and raise dues somewhat. But the benefits would be as follows — The total package at $45 would be significantly cheaper than the $61 it costs separately. Members would have access to the USUS libraries, MUSUS, the NewsLetter, the annual meeting, AND the journal for that price. There are 15,000 issues of the journal circulated. We can also expect that some other readers would find membership at $45 worthwhile, compared to $26 for the journal alone. The readership is a select group with an interest in Pascal and Modula-2. The increase in membership could offset the short-term loss of income. Increased utilization of MUSUS would also increase USUS income. And JPAM libraries may become available to USUS members more directly.

I believe that the members who have 'voted with their feet' are saying that USUS without a journal isn't worth the price of membership. So now I am asking a critical question of all members.

Is USUS <u>with</u> a journal what you want?

Would a package as proposed be acceptable?

Would members who have not renewed now renew?

We must have your answer. If there is a significant positive sentiment, we can proceed with negotiations immediately. If the sentiment is roundly negative, the matter will be dropped. To help you decide, we have arranged for all members to receive at least one free copy of JPAM. Review this copy, and see if you think you would like this journal as one of your membership benefits. If you did not receive a copy, contact one of the Board of Directors and we will try to get you one. We need all members' (and recent members') responses. We need to count 'votes' on this subject, so that we know how to proceed. Votes from members who did not renew will be especially important.

Send your response to the USUS post office box or to me on CompuServe (PPN: 75226,3643).

Orphan Software Project (June 1, 1989)
by Beverley Henderson

Well, I'm finally ready with another report on the Orphan Software Project. The responses have slowed to a stop and I have only two returns to give you this time. Both of them, however, include several programs.

E. H. Henderson of Coach and Camper Service in Bishop, CA has a Sage IV computer using UCSD IV.1 and runs five Timberline programs and a Pascal Development Program from Sage. Four of the Timberline products are accounting software and the other a spreadsheet. He says, "Timberline no longer writes programs which will run under the p-System." He also reports that the authors of the modular accounting package he uses dropped the p-System (BEFORE completing the accounts payable and inventory modules) and now use C.

Joseph Asling of Davis, CA uses a Sage II with p-System 4.13. His reported software includes Aladin Relational Data Base by ADI America; Logicalc by Software Products International; PDBase by IOTC; a Timberline spreadsheet and Menu7 by Micro R&D. He also has "a few odds and ends" including ACE v0.9. Is this the ASE editor or something else?

He is very interested in looking at and possibly exchanging software. I wonder how valid his assumption that "no-one would get after us for copying the truly orphaned software" will prove to be. At any rate, he even offers to help with the project. We accept, I'm sure. I only hope he is not put off by the delay in reporting his letter. His address is 17 W. Lakeside Place, Davis, CA 95616. The telephone is (916)758-5667.

He also asks where El Granada is, as the ZIP code indicates it cannot be too far from Davis. Just in case others are also curious, El Granada is a small town on Highway 1 south of San Francisco between the end of Devil's Slide and Half Moon Bay. In fact, for those of you who have travelled the Coast Highway in this area, it is opposite the harbor and tracking station at Princeton. On the ocean side of the Highway the town is Princeton — on the other side it is El Granada. While there is not much to see right along the highway, the town extends up into the hills with a population density that is appalling and growing every day. Most of the streets are not paved and the house prices are astronomical for the conditions.

So much for the real estate and geography section. I will mail Joseph several copies of the Software Information Request Form and ask Frank Lawyer to be in touch regarding his offer of assistance. Since I am always going to be unable to devote much time to this project during the first four months of the year (my primary profession is tax preparation — enough said), I would be very grateful to have a backup then. Besides, all the help we can get will still leave this a very large job.

The next stage in the project is to chose the form in which we want to record the data so that it can be regularly updated and readily accessible to USUS members. This seems to require that we use a p-System data base (and we'd prefer to anyway, right?). Has anyone out there any suggestions for the program that would handle a fairly complex, fairly large database, is p-System based and (ha ha) still supported somehow? It would need to have the capacity to sort on different fields for report purposes as we would like to produce the list by vendor, by program name and by program purpose (i.e., spreadsheet, accounting, etc.). Please let us have some input on this.

Another plea for participation: Do any of you know others who, while not members of USUS, are p-System enthusiasts? If so, would they fill out our questionnaire if we sent them one? If you will give me names and addresses for such people, I'll send them the form with an explanatory letter and we may be able to expand our information. Of course, I should also include some information about USUS and an invitation to join!

Yet another plea: If you haven't sent in the form for your own p-System holdings, PLEASE do so now. The more we have when we design the data base, the less revision of the format will have to be done later. And thank you to everyone who has taken the time to help so far. We'll probably be knocking at your door again someday.

We Get Letters...

I received the following letter from ***Greg Jahn****, Systems Programmer, Data Center, Boise State University, Boise, Idaho 83725; (208)385-3891:*

"We are looking for a Modula-2 compiler for our system, and I was hoping that you may be able to help us in this pursuit. I have tried to get information from several network sources but have so far been unsuccessful.

We are using an AT&T 3B2 Model 600 with Unix Sys V, rel 3.1. We would be using Modula-2 for general instruction in our Computer Science courses, and a basic "vanilla" version would be acceptable. Since we are a University, cost is a significant factor in our consideration."

I also received a letter from ***Bernard Perrenoud****, Ingénierur E.T.S, 5, chemin de Châtenaya, 2013 Colombier, Switzerland. Besides volunteering to look at the Rainbow manuals, he says that he is interested finding contacts in the USA who are interested in computer science, aviation, aircraft and ship models.*

Administrator Says
by Hays Busch

I'm going to start this out with some software that has nothing to do with UCSD Pascal! Instead it is written (I'm told.) in Turbo Pascal and assembly, and is designed to run on IBM PC's. It's called **Finance Manager II**, and is "user supported software". A fully functional disk was sent to USUS for preview and I tried it... and then I bought it!

I'm pretty good with figures and can run a spreadsheet with reasonable insight, but a "bookkeeper" I'm not! So when I can use a program, and without going to school set up a set of books, the program's got to be pretty good! For home use, almost all data comes directly from your checkbook. You can simply start putting the checks in and allocating them to whatever expense categories you want, and lo and behold, you have a home bookkeeping system! I'm sure it would be equally easy to use for small to medium size businesses.

You can test the program with the General Ledger Module, and this is the one I can send to any USUS member for $2.00 (charge covers disk and mailing cost). From there on, you're free to buy your own registered copy from **Hooper International**, or decide you're not interested. I wound up buying the Reconciliation and Financial Utilities Modules in addition to the General Ledger Module. If you want them, you can also get Modules for Payables, Receivables and Payroll.

The entire program is menu driven and almost self explanatory. (My only problem now is which is a debit and which a credit!) Documentation is excellent and explains some bookkeeping facts, service is excellent and the folks are very friendly and easy to talk to. I am thoroughly impressed! If you want to try this, drop me a note and enclose a check or money

Treasurer's Report (May 1989)
by Robert E. Clark, Treasurer

Bank Balance	$5,594.71	04-30-89
Income - May 1989		
Dues:		(new/renew)
Student	25.00	1/0
General	465.00	2/10
Professional	0.00	0/0
Institutional	0.00	0/0
Other Income:		
Library fees	13.00	
CIS	18.45	
Misc.	30.00	
Total Income:	$551.45	
Expenses - May 1989		
Administrator:		
CIS	10.06	
Telephone	55.67	
Photocopies	8.25	
Postage	38.19	
Supplies	88.55	
Other:		
Mail from La Jolla	10.02	
Library Distributor	8.00	
Bank charge	1.00	
Total Expenses	$219.74	
Bank Balance	$5,926.42	05-31-89

Treasurer's Report (June 1989)
by Robert E. Clark, Treasurer

Bank Balance	$5,926.42	05-31-89
Income - June 1989		
Dues:		(new/renew)
Student	0.00	0/0
General	700.00	0/20
Professional	100.00	0/1
Institutional	0.00	0/0
Other Income:		
Library fees	26.00	
CIS	25.91	
Misc.	20.00	
Total Income:	$871.91	
Expenses - June 1989		
Administrator:		
CIS	83.37	
Telephone	38.82	
Photocopies	10.00	
Postage	77.09	
Supplies	31.99	
Other:		
Mail from La Jolla	1.95	
Printing NL (2)	404.60	
Mailing NL (2)	781.28	
NL supplies	2.67	
Delaware Rep fee	70.00	
Bank charge	1.00	
Total Expenses	$1502.77	
Bank Balance	$5295.56	06-30-89

order for $2.00 made out to USUS, Incorporated.

Got a nice note from **Ed Livingston** in North Carolina. He's volunteering to be State Chairman for USUS down there. By the time you read this, we will have decided what we can do to reactivate some dormant USUS members in his state. Ed also thinks USUS should continue to work with JPAM on some basis since it could reduce our publishing cost and get us a wider audience.

When Arley Dealey opted to drop his MUSUS SysOp chores, I got to be the head SysOp for a short time. (Now, I'm helping maintain the MUSUS user listings.) All in all it was a very interesting experience.

Now that MUSUS is an "open" Forum, we've had 142 new people "visit" the Forum in a single month! Many are interested in Modula 2 and Pascal and some of these are becoming "regulars". One does not have to be a member of USUS to use MUSUS, but it is interesting that at least 19 active USUS members have not used the Forum since the first of this year. I can also identify 22 inactive USUS members who use the Forum fairly regularly!

But perhaps the most interesting statistic... about 210 USUS members may not have a CompuServe PPN and thus can not even look at MUSUS! If you are one of these and would like to try CompuServe, I can help you. I have a CompuServe brochure that tells all about the service, gives you a temporary PPN and some "free" connect time to test it all out. If you want one, drop me a note. There is no charge.

Harry Baya is the new SysOp. He is doing a complete revamp of MUSUS to bring the Libraries and Message sections up to date and make them more interesting. You can see some progress now and more changes are in the offing.

Board of Directors Minutes (June 13, 1989)
by Samuel B. Bassett

MINUTES OF THE SPECIAL MEETING OF THE BOARD OF DIRECTORS OF USUS, INC., HELD IN ROOM 1 OF THE MUSUS FORUM TELECONFERENCING FACILITY ON THE COMPUSERVE INFORMATION SERVICE, BEGINNING AT 10:10 PM EST JUNE 13, 1989.

Present at the meeting were:

User ID	Name
76314,1364	Sam'l Bassett, Board Chaircritter
73447,2754	Henry Baumgarten, Board Member
75226,3643	A. Robert Spitzer, Board Member
76703,500	Eliakim Willner, Board Member
72135,1667	Harry Baya, Primary SysOp
72747,3126	Robert Clark, Treasurer
71515,447	Alex Kleider, President
73007,173	William Smith, Assistant SysOp, NewsLetter Editor

Matters dealt with were:

<u>Nominating Committee for BoD Elections</u>

Harry Baya nominated Frank Lawyer, who wasn't present to defend himself, to be a member of the Nominating Committee. All present assented.

Alex Kleider said that he'd be happy to serve on the Committee, if Henry Baugartner would, too. Henry said he'd be willing to try, but that he would be out of town on vacation for a few weeks, in a part of Minnesota that doesn't have electricity.

Everyone thought this was fine, especially after Eli found out that he was not being volunteered for anything.

William Smith announced that the deadline for nominations is early September, so that ballots can go out that month in the NewsLetter, get back, and be counted in time to install new board members around the first of the year.

The Chaircritter stated <u>loudly</u> the he is <u>NOT</u> running for re-election.

<u>Annual Meeting</u>

When asked by the Chaircritter, William Smith said that there had been no response to the note in the last NewsLetter, asking for member input on the General Meeting.

Harry asked whether any location — East Coast, West Coast, Midwest, or Florida would be acceptable for a meeting. He was assured it would.

Bob Spitzer proposed Detroit because it is an airline hub, he thinks he might be able to persuade the University he works for to provide free meeting space, hotels are relatively cheap, and he has a 2-mile long lake in his backyard for recreation purposes. He also opined that we should piggy-back the USUS meeting with some other organization.

Harry volunteered to work with Frank Lawyer on finding East Coast sites away from the high-price zone. Bob Spitzer agreed to check on Detroit arrangements.

Alex suggested that Harry and Bob Spitzer be appointed to the Meeting Committee and report back to the Board later.

Henry asked about the advisability of involving JPAM in the planning or management, and Alex said he'd talk to them.

Bob Spitzer suggested involving Eli Willner, who was willing, so long as he didn't have anything to do with facilities, management, or arrangements. He also said that Liaison and Pecan would NOT be attending any shows this year or next.

Alex moved that all meeting business be remanded to the Meeting Committee, and that the BoD Meeting adjourn. Bob Clark seconded the motion,and it was carried unanimously.

<u>Next Meeting</u>

The Board agreed to again at 7 PM PST / 8 PM MST / 9 PM CST / 10 PM EST July 11, 1989 in Room 1 of the MUSUS conference facility

The Special Meeting of the Board on CompuServe was adjourned at 11:09 PM EST on June 13, 1989.

Minutes submitted by:
Samuel B. Bassett

The Chaircritter's Soapbox

by Sam'l Bassett

You may have noticed that the Nominating Committee for the upcoming Board of Directors election and the Meeting Committee are staffed by the "old familiar faces" (actually names) that you've been seeing over and over in the Board Minutes you see here in the NewsLetter.

In a lot of organizations, that would mean that there is a tight clique who is intent on keeping power — not so in USUS. What it means here is that there is nobody else that we can trick, coerce, finagle, or tempt into doing the work. We did manage to get an "outsider" as Secretary — after much searching and jawboning — **Howard Sweet**.

We would VERY much like to get some new blood and new bodies into the administration of this outfit, but if you keep on hiding your light under bushel baskets, that leaves us tired old hacks to do the work — how about giving us a break, and taking on some of it???

You'd have our undying gratitude . . . .

Apple SIG Doings

by Frank Lawyer

New Members...

Welcome to new members **Harry McMackin** and **Fred Nelson**. Please let me know if there is any help you need. In the case of Harry, he has already sent a letter! Answers in progress.

Welcome Disks...

The welcome disks have been updated to 07/01/89. There were only minor changes in the files, but there were changes on the order form for library disks. Some older "archive" disks have been withdrawn, since the programs have been converted to run on the Apple2 series of computers, and are now available on the Apple specific volumes. If you intend to order library disks, please get new welcome disks and use the order form on them, or send to me for a current order form.

Library Disks...

There is one new library volume this time, APP2U05, which is a utility volume. By next time I can see that there will be at least one more volume available. I seem to be falling behind with conversion and review efforts. As always, help would be gleefully accepted. My original goal was to convert all the old disks to new Apple specific disks in an 18-24 month period. It looks like I better work harder.

Volume APP2U02 has been updated with a new program, and anyone who purchased APP2U02 during 1989 is therefore entitled to a free update. Just send me your original disk and I will update it for you. The same applies to APP2U04, which has updated dictionaries for the SPELLER program. Several files were added to APP2I06. Listings, as usual, should be found elsewhere in this issue, by the grace of our Editor.

I haven't mentioned it in a while, so it probably bears repeating. If you order the new series (APP2XXX) disks, I expect them to work on your equipment, provided you use Apple v1.3. If they don't, or you have difficulty compiling etc. then I will help you if you call or write me and tell me what problem you are having. I have compiled and tested all the programs on the disks now, and although I can't say I have tested every feature of every program, I want the programs to work correctly. In general, I am willing to fix it if it is broken.

The Apple specific library volumes currently available are:

Information volumes:
APP2I01, APP2I02, APP2I03, APP2I04, APP2I05, APP2I06

Utilities:
APP2U01, APP2U02, APP2U03, APP2U04, APP2U05

Games:
APP2G01, APP2G03, APP2G04

In Retrospect...

It's hard for me to believe that this column is a year old now. As a penance, I forced myself to read all my old columns from the past year. This is like being locked in a room with a life insurance salesman! I have had fun, although sometimes it seemed that I would never have enough material to fill the next column. I have been in contact with a number of you, and have gotten letters and contributions from a reasonable number (who counts?). Lately I have fallen behind on my correspondence with some of you out there, so if you wrote me a letter two months ago, and didn't hear yet, don't give up!

USUS has just passed (June) its ninth anniversary, and it looks to me like we will make ten. Somebody out there should start thinking of stealing this job away from me.

Utilities...

When I download messages from MUSUS, I usually do a small bit of editing, then save the files as a historical record, in case I want to find something later. From what people on MUSUS have said in the past, others do this also. It tends to fill disks up pretty rapidly, but there is some valuable information there. The real problem is trying to find what you want, when you remember that the answer to someones question is out there, or you want to review all the banter regarding issues of "public domain" that we batted around over the months. Or maybe you want to find all the messages from a particular person. Over time, I have experimented with various methods, but they all involved extra work or my part. It's easy to find a key word or phrase with the editor F(ind command, but when you have several hundred files contained on a number of disks, it becomes daunting. What I really needed was a program that could locate items of interest with as little manual intervention as possible. I put it on my "write someday" list.

Well, I recently remembered seeing an article by **Jon Bondy** in an old USUS News and Report (#7) where he had the same problem, and he wrote a program to solve it. SCANNER will run through all the files on a volume, looking for the set of strings that you enter into a file called SCAN.DATA.TEXT, and will display the file name and the text portion that contains the string. I just knew I wouldn't have to re-invent the wheel. As a beneficial side-effect, SCANNER locates the desired strings anywhere on the volume, so it makes it handy to find where that mistakenly erased file is hiding out. I put SCANNER on volume APP2U02, in case you need it too. Thanks Jon!

I'm going to use SCANNER to help me with the library review. There are a lot of programs out there, and even though I know a pretty fair amount about what IS in the library, I get questions that require some browsing through the disks. This will help me locate programs much more quickly.

Programming Project Ideas...

I have asked the kindly editor to reprint a column by **Robert Platt,** a USUS member, author of a book on Modula-2 and Editor of the "Perfect Pascal Programs". Bob writes a monthly column for the Washington Apple Pi Journal. In this column he has several interesting ideas on Pascal programming projects, and he asks the WAP members to show their interest. Since the Apple SIG members have pretty much identical interests, I thought we might get interested also.

I agree with all the suggestions that Bob makes for programming projects, but the two which are most interesting to me are the "glue" routines, and the project to support various fonts. If you are at all interested in these as co-operative programming projects, please drop me a line, or give me a call.

Frank Lawyer, 126 Demott Lane, Somerset, NJ 08873; (201) 828-3616

"Pascal News" (reprint from WAP Journal)
by Robert Platt

The following is the text of ***Robert Platt's*** *Pascal column, "Pascal News", which appeared in the March 1989 issue of the WAP Journal, the monthly magazine of the Washington Apple Pi. It is reproduced with permission granted by their general provision for computer clubs to reprint their articles.*

II GS Wish List

At present, an Apple Pascal programmer seeking to port p-System programs from the IIe to the IIgs faces a serious dilemma. On the one hand, he can stay with the p-System. Pecan Software sells a p-System hosted under ProDOS 16. Apple Pascal 1.3 will also run on the IIgs {in 2e mode - fl}. Yet, neither system permits direct calls to the IIgs ROM tools. On the other hand, new Pascal programming environments have appeared. Apple's APW features Pascal compilers from TML and ORCA/M. Stand-alone Pascal systems are also available from TML and ORCA which feature their own "shells". Such shells make debugging much easier and eliminate the delays caused by disk accesses between program editing and compilation. Needless to say, these programming environments support fully the IIgs ROM tools. Of course, converting UCSD Pascal source to run under these environments is a little like pulling teeth. There are two possible solutions to this problem, and hence my wish list:

- Someone could write a translator to convert UCSD Pascal source into TML or ORCA Pascal source files.
- A set of glue routines written under the p-System Assembler which would permit calls to the IIgs ROMs should be fairly straightforward. These were promised by Pecan when they first released the ProDOS 16 version of the p-System. We need them!
- One of the neatest features on the IIgs is its support of fonts. The font files on the IIgs are similar to those on the Macintosh. The Apple Pascal p-System has traditionally supported bit-mapped fonts under the Apple II's hi-res graphics. The only font supported was stored in a file called SYSTEM.CHARSET. An interesting programming project would be to modify the WSTRING and WCHAR routines in the APPLESTUFF library to display fonts from any IIgs font file rather than just from SYSTEM.CHARSET.

Perhaps one of the above wish list items would make an interesting group programming project for the PIG {Pascal Interest Group of WAP}. How about it?

Note for IIc Plus owners

Apple Pascal 1.3 is available from APDA on a 3.5" disk format. Don't miss out on this programming bargain. I would be interested in hearing from a IIc plus owner who would be willing to run benchmarks to see whether performance improves when running Apple Pascal on this new accelerated machine.

TI SIG News

by Ken Hamai

As promised, more disk drive information. This time we go to the more exotic stuff, what to do with 4 disk drives, and Myarc disk controllers. Let's start off with Myarc.

The latest rage from Myarc is the hard disk controller. With your TI and this controller, you can run your Pascal system boot disk (Unit #4) from the hard drive. This tip came from new member, **Ed Livingston**, of Lenoir, NC, and was published in the Micropendium. Ed says, to use the hard disk, put all of your system files, ie. editor, filer, compiler, etc., on this one emulate file and your system will then recognize it as Unit #4. Before you power up the p-System, do a call MDM through Basic and turn on your DSK1 emulate file. When the system initializes, it will use the emulate file as Unit #4. Your disk drive DSK1 then becomes Unit #5, and DSK2 becomes Unit #9. Thanks for the tip Ed! By the way, Ed also says he is willing to help anyone who wants help with firing up the TI p-System and invites you to write him at 244 Walt Arney Rd., Lenoir, NC, 28645.

Working independently, new member **Frank Aylstock** of Placentia, CA tells me that he set his emulate file up for quad density emulation, which gave him 1440 blocks! This is 8 times larger than what the standard TI format was back in 1983. He said with the hard disk, the compiler just zooms through compilation! Another good excuse to buy that hard disk controller!

The hard disk controller can also use floppies, but let's think of more exotica... Using the 3-1/2 inch disk drives! These drives now cost anywhere from $35 to $85 and they are almost a plug in for the TI. Many are now sold with face plates and adapter cables so that they will fit where you once had your half height 5-1/4. With the Myarc controller, you can use this type of drive in 80 track, double density, which gives you 2880 sectors, or 1440 blocks. If you can find them, you can also use the 80 track 5-1/4 disk drives and get the same 1440 blocks.

What about the guy with the Myarc floppy disk controller. He's got a different problem. Myarc, in their infinite wisdom(?) decided that their standard format for double density was to be 16 sectors per track. So their disk controller default is 16 sectors. If you used their optional 18 sectors per track, the controller placed this information on a track 0 of the disk. Just so happens that the p-System doesn't ever look at this track! So now what happens is the Myarc system assumes that you have a 16 sector disk!

This is fine if the disk is truly 16 sector. But what if you were a rational man and wanted to use the maximum available, 18 sectors? Well, you had to get the Myarc controller to read track 0 of each disk in each of the disk drives. This was easy enough to do with a little Pascal startup program, but you still had the problem with the boot disk in Unit #4. It had to remain at 16 sector because remember at boot up, the controller hasn't been told yet that you have the 18 sector disks! This can be a real pain in the you know what!

Jerry Coffey figured out how to fix this problem with a trick startup disk that had the startup file crammed into the 16 sector portion of a specially formatted disk. If you need this information, send me $3.00 for the disk, I take out postage and disk cost and give USUS the change, and you get the disk. Or leave a message for Jerry Coffey on CompuServe. He can tell you what file you need to download.

Ok, one more problem to cover... What to do with DSK4? Well, believe it or not, that has also been taken care of with a system startup file. The information comes from Anders Persson of the Swedish Pascal Interest Group. He has a startup file with an assembly program that pokes in the codes needed for the p-System to recognize Unit #10 as a disk drive. This information is also available on CompuServe. Or you can get the disks from me, Jerry Coffey or the Boston Computer Society TI group. If you want the whole package, which consists of the Myarc file, plus all the info on the Anders Persson disk, send me $5 and ask for the Myarc and Swedish disks. Of course, any money left over after postage and disk costs go to USUS.

Well, that's it for disk drives. While I'm at it, I also wanted to tell you about what is going on with the USUS library. So far, I'm down to about volume 7 and have found several programs that compile, many that don't, and out of those that do, only a few are really useful. This is the equivalent of what we had with the TI back in 1982 when people were only hacking away with their consoles and tape recorders writing prime number generators and cpu speed tests. Well, to tell you the truth, that sucks! Therefore, I will be compiling all of this together and will be making up special disks for the TI only, and try to separate the trash from the good stuff. When I have enough for a 640 block Volume, I'll let you know. I'm about half way there now.

In the meantime, the TI Pascal SIG organized by the Brea Users Group here in California is cooking along. We even get homework (Ugh!). And thanks to **Bob August**, we have a slick LISTER program that lists text files to printer or screen, along with line numbers (optional). Bob also discovered another odd thing. He wrote a program that works fine if it's named SYSTEM.WRK.CODE, but change that name to any other name and the system comes up with the dreaded STACK OVERFLOW - REBOOT! I don't know what's causing it yet. Must be something about how the system allocates memory. It must be doing it differently when running a SYSTEM.WRK.CODE file versus any other named file. I guess this is almost the equivalent of the problem where you have a Basic program that is too big to run from disk but works fine from tape. That extra memory used by the disk buffer cuts back the overall program size and array sizes. If anybody has any ideas what the problem is, please let me know. Bob says that the program is a name and address program that sets up an array in memory. Looks like the array is a little too big, Bob.

One more thing, if you're new to the p-code system and want to know more about how it works, member **Ron Williams** of the Boston Computer Society writes excellent articles covering various aspects of Pascal programming. All of his past articles are available on disk. Write Ron at 14 East Street, Avon, MA 02322.

I got a call the other day from member, Ed Livingston himself, in response to my last USUS article. Ed called to let me know that I had been incorrect in saying that there were no other compilers for the TI p-code card besides Pascal. He said that he had a FORTRAN 77 compiler and it ran on the TI! Well, well, I don't know were this came from, but I'm certain it wasn't from TI. In any case, Ed tells me that the compiler works fine and is closer to being like the "real" FORTRAN than the other TI version, 99-FORTRAN. Only problem, he hasn't been able to get the compiler to accept the TI units which access the graphics, sprites and sound. It looks like sometime in the past, someone (maybe it was Batman or the Lone Ranger) ported the compiler over to the TI from another system. If this is workable, maybe someday, we can get a BASIC compiler also ported over. Somehow, I get the feeling that this is on the verge of being illegal. Maybe this time, we'll let Ollie do it. Or maybe we'll let Ollie's boss do it, whoever THAT may be. HE can get away with anything!

From **Andrew Becker** in Jamaica, NY comes more news of his success with connecting a video terminal to the TI and having the p-System run the terminal. This is the poor man's answer to the 80 column bugaboo. So, if you can get your hands on a terminal, you too can have 80 column screens when running the p-System. Andy tells me that he wants to donate his work to the USUS library to let the other people try it. He has files for running a VT52 and ADM3A terminals. If you have another type of terminal that does not have either emulation, the work that Andy did would still be useful to you. As I understand it, you would need to change the MISCINFO and GOTOXY files to your terminal. One major problem, according to Andy is the long screen update time even with the terminal and RS232 set at 9600 baud. He guesses that it is due to the TI's slow method of redirecting output in the p-System and the p-code interpreter not helping out any. We have a copy of Andy's work in the SIG library. If you want a copy, send me $2.00 and tell me you want the BOOT.VT disk. I take out the cost of the disk and mailer and USUS gets the change. Andy, a great big thanks for your work!

Till next time, keep them p-cards warm.

Ken Hamai, 11508 Mollyknoll Ave., Whittier, CA 90604; (213) 943-1194

UCSD-like string operations in Modula-2

by A. Robert Spitzer MD
Wayne State University

Modula-2 is a more powerful, flexible language than Pascal, created by Niklaus Wirth to be used as a "real" programming language, by experienced programmers. The concept of Pascal was originally a teaching language, and many of the restraints that experienced programmers find frustrating are relaxed in Modula-2.

Modula-2 was also designed to be much more portable that Pascal. Actually, Pascal is one of the least portable languages around. (UCSD Pascal is no exception — it gets around the problem by creating its own environment on every machine it runs on, and there are NO UCSD compilers outside of p-System. If you want a really portable language, use FORTRAN-77!) One of the major reasons for this is the extensive definitions for I/O in Pascal. Wirth sought to address this problem by (paradoxically) eliminating from the language definition anything superfluous or possibly machine dependant — with considerable success.

This has created a situation in which many novice programmers, who are reaching some of the limits of Pascal, shy away from the switch because of the gap that must be bridged. One such gap is a significant difference in string handling. Modula-2 does not contain any "string" type, nor any string manipulation procedures. Rather, it merely supports ARRAYs of various types, including "ARRAY OF CHAR." An array of CHAR is merely an array of characters. There is one common convention, namely that (scanning from left to right), the presence of null character (CHR(0)) indicates the end of meaningful content in the array. This is in contrast to UCSD Pascal (and most other Pascals), in which the leading character indicates the length of the string. Note that a Modula-2 string can be longer than 255 characters — it can be as long as you want!

I provide the following module to help individuals bridge the gap from Pascal to Modula-2; in the process, one can learn how to manipulate Modula-2 "strings"; and one can develop an appreciation for the power of Modula-2, namely that "low level" operations such as these are <u>entirely legal</u> and do not require any sort of tricks, do not require trick records, do not require range- or type-checking suppression, etc. This module should also illustrate that once one has mastered the language a bit, developing modules to support many of the seemingly mystical and complex functions that are "built in" to other languages (and seem to be "missing" from Modula-2) is actually quite trivial.

<u>Operational descriptions of procedures exported by StringOps.</u>

<length> returns the length of the meaningful contents of an ARRAY OF CHAR. This is not

the array length, which may be larger. The length is 1-based, i.e "0" means no meaningful characters are in the string. All characters, including non-printing characters, are potentially valid, except the Modula-defined NULL terminator. If the array is entirely full, HIGH(s)+1 is returned (this is the correct value).

<concat> concatenates contents of "source" onto the end of "dest".

<scan> searches "s", starting at position "start", until "substr" is found, then returns its position. "start" is useful for repeated sequential searches, or part-string searches. If the returned position is greater than HIGH(s), the substring was not found. For short strings, the algorithm is trivial; for 'long' strings (a compile-constant definition), the search uses a high speed Boyer-Moore algorithm.

<insert> inserts "substr" into "s" at position "index".

<delete> removes "size" characters from position "index" in s.

<strCompare> does a lexical comparison, using whatever numbering base is inherent to your machine (usually ASCII), and returns an enumerated type indicating the result of the comparison operation.

<xtract> copies a substring of length "size", beginning at position "start", from string "from" to string "result". This is useful when building a string parser, manipulating file and volume names, etc.

<copy> copies "from" into "to".

Some notes on error checking, and a bit of philosophy.

"When there is no error, there is no need for error checking." — Gary Knott 1986

You may notice in my code that error checking is minimal or non-existent. For example, if I hit the end of an array, my code just stops (RETURNs), with no error message. There are some very good reasons for this.

#1) It is impossible to bypass low level error checking higher up. Specifically, error checking is slow. Often, error checking actually takes more time than the code itself. You can NEVER regain this speed in you higher level code. Error checking code is executed even if there is NO error. Often, checking is slower if there is no error than when there is an error, because all characters are checked, whereas if an error is detected early-on, processing is aborted immediately. This may seem trivial at first, but when you are processing a 1000-page document it can save hours of processing time.

Theorem 1. It is faster not to check for errors.

#2) To check for errors, you must know what is an error. At first this seems trivial. But remember again, that the definition of an error low down can never be changed by code higher up. For example, lets say you want to write a file-transfer program, and your low-level code decides to filter all non-printing characters from strings, because these characters "must be an error". With that kind of low level module, you will NEVER be able to decide that certain control characters are legitimate and should be sent. What if your file contains <Ctrl-Z>s, but they are NOT end-of-file markers!

Theorem 2. It is hard to know what the definition of an error will be.

#3) If you check for errors, you must know what to do about them. OK, you found an error, what now? Often, the decision is made in the low level code. This results in horrible disasters that can't be fixed higher up. For example, try to use the p-System string-handling procedures to read input from the user. If you try to read an integer, and the user types an invalid character, the system blows you entire program out of the water, and prints an obscure message on the screen. Imagine this happening inside some complex database program you wrote, and you application dies with open files, partially updated records, invalid temporary pointers, etc.. You could kill the entire database! And even if that doesn't happen, imagine error messages popping up at random all over you nicely formatted, windowed screen. Not good. What if you would rather log errors to a file? No, you want to trap the errors and handle them higher up!

Theorem 3. Decisions about how to handle an error must be made high up. Corrolary 3. Error reports should NEVER go to the screen. The should be passed upstairs in the program for handling.

Therefore, philosophically, error checking is really always a HIGH level operation, as close to the user interface as possible.

Further consider Spitzer's rule of errors.

"Errors are always your fault."
Corrolary: "Errors should not happen".

There are two kinds of (software, not hardware faults) errors. An error condition might arise because of a bug in your program. This is clearly the programmers fault. One uses error checking code to test for these during development. Once development is complete, and all the bugs are gone, production code does not need error checking.

The only other error that can occur is because of "incorrect data". For example, the user may have entered "July 2, 1986" when the input required was of type "money". This could cause incorrect processing at a lower level. If that happens, it is also the programmers fault. If it is possible for a data error to occur, YOU should check that before calling the low level procedures (for reasons I listed above). Once you have decided that the data is correct up front (at a very high level), there is never any need to waste time checking it a each lower level. For example, if an operation will require a division somewhere, you better check your divisor for zero before you call that procedure.

For those situations where these two cases do not apply, I strongly urge that your procedure merely have a safety-default, or return a simple "error code", which can be processed higher up, for the aforementioned reasons (2 & 3).

If this has encouraged you to switch rather than fight, I also hang out on MUSUS, and try to help with Modula-2 language questions (sorry, but I know nothing about any of the IBM or Apple specific implementations, as regards their machine or library-specific details).

Regards and happy coding. Bob Spitzer.

```
DEFINITION MODULE StringOps;
(* Copyright (c) 1987
    A. Robert Spitzer MD
       Change log:      18 Nov 87
        4 Mar 1987      Created.
*)
(* 5/23/89: may be published by USUS
for use by members only.  Compiled code
may used by members. Source may be
redistributed by USUS only.  All other
rights reserved. *)
FROM Globals IMPORT relative;
(*
TYPE relative = (lessThan, equals,
                 greaterThan,
                 lessOrEqual,
                 greaterOrEqual);
*)
EXPORT QUALIFIED length,
                 concat,
                 scan,
                 insert,
                 delete,
                 strCompare,
                 xtract,
                 copy;
PROCEDURE length
        (s:ARRAY OF CHAR):CARDINAL;
PROCEDURE concat
        (VAR dest   : ARRAY OF CHAR;
             source : ARRAY OF CHAR);
PROCEDURE scan
        (s,
         substr : ARRAY OF CHAR;
         start  : CARDINAL):CARDINAL;
PROCEDURE insert
        (substr : ARRAY OF CHAR;
         VAR  s : ARRAY OF CHAR;
          index : CARDINAL);
PROCEDURE delete
        (VAR  s : ARRAY OF CHAR;
          index,
           size : CARDINAL);
PROCEDURE strCompare
        (s1,
         s2 : ARRAY OF CHAR
        ):relative;
PROCEDURE xtract
        (from : ARRAY OF CHAR;
        start,
         size : CARDINAL;
    VAR result : ARRAY OF CHAR);
PROCEDURE copy
        (VAR from,
               to : ARRAY OF CHAR);
END StringOps.
```

```
IMPLEMENTATION MODULE StringOps;
 (* Copyright (c) 1987
      A. Robert Spitzer MD
    Change log: 21 Feb 89
    21 Feb 89   corrected 'copy' to
                deliver the correct
                number of bytes and a
                null string.
    25 May 87   Included straight and B-
                M string search.
    4 Mar 87    Created.
*)
FROM Globals IMPORT relative;
(*
TYPE relative = (lessThan, equals,
                 greaterThan,
                 lessOrEqual,
                 greaterOrEqual);
*)
CONST  null = CHR(0);

PROCEDURE smaller
          (a,b:CARDINAL):CARDINAL;
BEGIN
IF b < a THEN RETURN b ELSE RETURN a
END
END smaller;

PROCEDURE length
          (s:ARRAY OF CHAR):CARDINAL;
VAR i : CARDINAL;
BEGIN
FOR i:= 0 TO HIGH(s)
DO  IF   s[i] = null
    THEN RETURN i
    END
END; (* for *)
RETURN (HIGH(s) + 1)
END length;

PROCEDURE concat
          (VAR dest  : ARRAY OF CHAR;
               source : ARRAY OF CHAR);
VAR i,j,sourceLen : CARDINAL;
BEGIN
i:= length(dest);
sourceLen:= length(source);
j:= 0;
WHILE (i <= HIGH(dest)) &
      (j <= sourceLen)
DO    dest[i]:= source[j];
      INC (i);
      INC (j)
END; (* while *)
IF   i <= HIGH(dest)
THEN dest[i]:= null
END  (* if *)
END concat;
```

```
PROCEDURE scan
          (s,
      substr : ARRAY OF CHAR;
       start : CARDINAL):CARDINAL;
(* search procedures modified from
   Wirth:Algorithms & Data Structures
   1986
*)
CONST
   longStr = 64; (* dividing line
   between straight and B-M search *)
VAR
   i,j,k,
   lenS, lenSub : CARDINAL;
   ch : CHAR;
   d : ARRAY[0C..177C] OF CARDINAL;
BEGIN
lenS:= length(s);
IF start <= lenS THEN
   lenSub:= length(substr);
   IF (lenS - start) > longStr THEN
   (* long string,
        do Boyer-Moore search *)
      FOR ch:= 0C TO 177C DO
         d[ch]:= lenSub
      END;
      FOR j:= 0 TO lenSub-2 DO
         d[substr[j]]:= lenSub-j-1
      END;
      i:= lenSub + start;
      j:= lenSub;
      WHILE (j > 0) & (i < lenS) DO
         j:= lenSub;
         k:= i;
         WHILE (j>0) &
              (s[k-1] = substr[j-1]) DO
            k:= k-1; j:= j-1
         END;
         i:= i + d[s[i-1]]
      END; (* while j>0 *)
      IF j = 0 THEN RETURN k
      ELSE RETURN lenS
      END
   ELSE (* short string, just do
           straight string search *)
      i:= start;
      REPEAT
         j:= 0;
         WHILE (j < lenSub) & (s[i+j] =
               substr[j]) DO
            INC(j)
         END;
         INC (i)
      UNTIL (j = lenSub) OR
            (i > (lenS-lenSub));
      IF (j = lenSub) THEN
         RETURN (i-1) (* position at
                         which found *)
      ELSE RETURN lenS (* not found,
               return end of string *)
      END
   END (* if lenS *)
ELSE (* can't start beyond end of
        string *)
   RETURN lenS
END (* if start *)
END scan;

PROCEDURE insert
          (substr : ARRAY OF CHAR;
           VAR  s : ARRAY OF CHAR;
            index : CARDINAL);
VAR i, lenS, lenSub : CARDINAL;
BEGIN
lenS:= length(s);
lenSub:= length(substr);
IF   (lenSub > 0) & (index <= lenS)
THEN IF   lenS = 0
     THEN IF lenSub <= HIGH(s)
          THEN s[lenSub]:= null
          END
     ELSE i:= lenS;
          WHILE (i > 0) & (i >= index)
          DO    IF   (i + lenSub) <=
                       HIGH(s)
                THEN s[i+lenSub]:= s[i]
                END;
                DEC (i)
          END; (* while *)
          IF   (index = 0) &
               (lenSub <= lenS)
          THEN s[lenSub]:= s[0]
          END
     END; (* if lenS *)
     FOR i:= 0 TO smaller(lenSub-1,
                  HIGH(s)-index)
     DO  s[index+i]:= substr[i]
     END (* for i *)
END (* if index<=lenS *)
END insert;

PROCEDURE delete
          (VAR  s : ARRAY OF CHAR;
            index,
             size : CARDINAL);
VAR i, j, lenS : CARDINAL;
BEGIN
lenS:= length(s);
i:= index;
WHILE (i + size) <= lenS
DO    s[i]:= s[i + size];
      INC(i)
END; (* while *)
IF   i <= HIGH(s)
THEN s[i]:= null
END
END delete;

PROCEDURE strCompare
          (s1,
           s2 : ARRAY OF CHAR
          ):relative;
VAR i, s1length,s2length : CARDINAL;
BEGIN
i:= 0;
s1length:= length(s1);
s2length:= length(s2);
WHILE (i+1 <= s1length) AND
      (i+1 <= s2length)
DO    IF   s1[i] > s2[i]
      THEN RETURN greaterThan
      ELSIF   s1[i] < s2[i]
         THEN RETURN lessThan
      END; (* if *)
      INC(i)
END; (* while *)
IF   s1length > s2length
THEN RETURN greaterThan
ELSE IF s1length < s2length
     THEN RETURN lessThan
     ELSE RETURN equals
     END
END (* if *)
END strCompare;

PROCEDURE xtract
          (from : ARRAY OF CHAR;
          start,
           size : CARDINAL;
     VAR result : ARRAY OF CHAR);
VAR i, j ,fromLen, toLen : CARDINAL;
BEGIN
fromLen:= length(from);
IF   start <= fromLen
THEN i:= start;
     j:= 0;
     WHILE (j < size) & (i <= fromLen)
         & (j <= HIGH(result))
     DO    result[j]:= from[i];
           INC (i);
           INC (j)
     END; (* while *)
     IF    j <= HIGH(result)
     THEN result[j]:= null
     END (* if j *)
END (* if start *)
END xtract;

PROCEDURE copy
          (VAR from,
                 to : ARRAY OF CHAR);
VAR i, len : CARDINAL;
BEGIN
len:= length(from);
IF   (len = 0) & (HIGH(to) > 0)
THEN to[0] := null;
     RETURN
END;
IF   HIGH(to) < len
THEN len:= HIGH(to) + 1
END;
```

```
FOR i:= 0 TO len - 1
DO  to[i]:= from[i]
END;
IF   len <= HIGH(to)
THEN to[len]:= null
END
END copy;

END StringOps.
```

WDS OpSys Unit

By William D. Smith

This is the unit which I used to interface to the operating system. No other units or programs of mine use any of the system units. Too bad the last statement isn't completely true. As you can see from last months column, F_Io_U does use the Kernel. Also since this is an ongoing project, I do have some programs (well one big one, my AltFiler program), written before this unit was developed which uses the DirInfo unit.

This unit was written so that I would not have to use all the system units in my other units or programs (i.e. isolate the my stuff from the system). I also only use a small number of items from each unit. When you are compiling a program which uses this unit instead of three or four system units, the compilation goes faster (slightly) since only one interface needs to be read.

The `NoBreak` function is used to disable the p-System break key. I use this in my startup program so that the user can not access the system until they first enter their password (or boot from a floppy).

The screen size procedures are use to get and set the screen size. This allows dynamic re-sizing of the screen without rebooting (after using Setup to change the screen size parameters). I use a Wyse-60 which can used several different heights (24, 25, 43 and 44) and two widths (80 and 132). My FlipScreen program lets the user choose what screen size they want, sets the terminal to that screen size and then changes the values maintained by the operating system. My terminal I/O unit (covered next NewsLetter, I think) reads the screen size from the system when it starts. This allows my programs to be written in a screen size independent way. Another use is to enlarge the screen (number of columns) when printing a report to the screen and then returning it to its original size.

The time and date procedures are used to get and set the system time. The volume procedures read or write the volume names choosen. `CopyFile` makes a copy of the named file. `Get_DirInfo` returns a list of files in the same manner as `D_DirList` of the `DirInfo` unit, except it doesn't return any pattern matching info. If only one item is returned, the time is return in the record correctly. If two or more item are returned, the time is set to the `NullTad .T`. I do this since the only way to get the timestamp is to open the file (a very slow process if the list is of any length at all).

`The Get_MyFile` name procedure was described in the Sept/Oct 88 issue of the NewsLetter.

Imported from `Kernel` are `IoRsltWd` an integer variable; `Alpha` a type of packed array [0..8] of char; `E_Rec_P` a pointer type which points to an `E_Rec`; `SysCom` a variable which points to the system communication area; and `UnitList` a variable of type `E_Rec_P` which points to the unit list.

Before this unit can be compiled, the interface sections of the `SysInfo`, `Transfer`, `ScreenOps`, `Wild` and `DirInfo` units needs changed. All references to a date record type (such as `D_DateRec`) need changed to `DateRec`. In a similar manner, time records (such as `D_TimeRec`) need changed to `TimeRec`, `string` needs changed to `Str_81` and long strings (such as `D_LongString`) need changed to `Str_255`. The type declarations need removed and a "`uses Glbs_U;`" needs added to the beginning of the interface. I used my advanced patch program to make these changes. If you don't have a patch program which allows you to insert and delete bytes, use Decode or LibMap to extract a copy of the interface, modify it so that it is compilable, make the changes and recompile it. Then use this interface when compiling this unit.

```
{ WDS operating sys interface unit [2.00] --- 07 Mar 88 } { |xjm$d|nx|f8|e|. }
{$Q+}
{$C (c)  William D. Smith  -- 1987 to 1988,  All rights reserved.            }
{ File:       OpSys_U.Text           Version 2.00    07 Mar 88

  Author:     William D. Smith       Phone: (619) 941-4452
              P.O. Box 1139          CIS: 73007,173
              Vista, CA  92083

  Notice:     The information in this document is the exclusive
              property of William D. Smith.  All rights reserved.
              Copyright (c) 1987 to 1989.

  System:     Power System version IV.2.2

  Compiler:   Power System Pascal Compiler

  Keywords:   WDS OpSys_U Operating System Interface Unit

  Description: WDS operating system interface unit.  This unit contains
              the procedures which interface to the operating system.

  Change log: (most recent first)

Date       Id   Vers  Comment
---------  ---  ----  -------
07 Mar 88  WDS  2.00  Move all the file access procedures to the F_Io_U.
07 Feb 88  WDS  1.10  Added Get_MyFile and Get_MyHelp.
06 Feb 88  WDS  1.09  Added Get_DirInf and DiDispose.
20 Oct 87  WDS  1.08  Added Vs_OpSys_U and its use.
16 Sep 87  WDS  1.07  Added Get_&Set_Filename, did some cleanup work.
31 Aug 87  WDS  1.06  Fixed for version IV.22.
16 Jul 87  WDS  1.05  Put in version control, added Get_, Set_LastByte.
19 Jun 87  WDS  1.04  Fixed so Tad need not be changed.
05 Jun 87  WDS  1.03  Rounded seconds to nearest minute in get TAD.
28 May 87  WDS  1.02  Added CopyFile.
27 May 87  WDS  1.01  Added more procedures and finished the others.
08 May 87  WDS  1.00  Started creating this unit.
}

{$I VERSION.TEXT}  { Declares conditional compilation flags }

unit OpSys_U;

interface {$ OpSys_U [2.00] 07 Mar 88 }

uses  Glbs_U;   { WDS globals unit }

const Vc_OpSys_U   = 8;   { 07 Mar 88 }
      Vs_OpSys_U   = 'OpSys_U';

type  FileKind     = (FkErr, FkVol, FkSvol, FkDir, FkCode, FkText, FkData);

      DiPtr        = ^DiRec;

      DiRec        = packed record  { 18 words }
                       Next     : DiPtr;
                       Kind     : FileKind;
                       Blocked  : boolean;
                       Filler   : 0..15;
                       UnitNum  : Byte;
                       Volume   : Str_7;
                       Title    : Str_15;
                       Size     : integer;
                       Start    : integer;  { if Kind = Vol, Start is # files }
                       Tad      : TadRec;
                     end { DiRec };

var   Vv_OpSys_U   : integer;
```

```
function NoBreak (DoWhat : OnOff) : boolean;
{ This function returns the current status of the p-System NoBreak variable.
  DoWhat is what to do before the status is read.  On turns on NoBreak (ie. no
  break allowed).  Off turns off NoBreak (ie. break allowed).  Toggle toggles
  NoBreak.  All other values of DoWhat return the status of NoBreak without
  changes.
}

function Get_Sc_Width : integer;
{ Get screen width. (1 based) }

function Get_Sc_Height : integer;
{ Get screen height. (1 based) }

procedure Set_Os_Sc_Size (H, W : integer);
{ Set op system screen size.  This procedure sets the height (H) and width (W)
  of the screen size maintained by the operating system.  Both H and W are one
  based.  If either is Null, it is not changed.
}

procedure Get_Sys_Tad (var Tad : TadRec);
{ Get system time and date. }

procedure Set_Sys_Tad (Tad : TadRec);
{ Set system time and date.  If either the time or date is null, it is not
  changed.
}

procedure Get_Pref_Vol (var S : Str_7);
{ Get prefix volume. }

procedure Set_Pref_Vol (S : Str_7);
{ Set prefix volume. }

procedure Get_Sys_Vol (var S : Str_7);
{ Get system volume. }

procedure CopyFile (Src, Dest : Str_23;  var Msg : integer);
{ Copy file.  This procedure copies the file named in Src to the destination
  Dest.  Msg is M_NoError or the last ioresult.  If Dest exists, it is purged.
}

function Get_DirInfo (     Src : Str_255;
                       var Ptr : DiPtr;
                       var Msg : integer) : boolean;
{ Get directory information.  This function returns the directory information
  of the file(s) named by Src.  The file(s) need not be open.  The function
  returns true if Msg = M_NoError, otherwise it returns false.
}

procedure DiDispose (var Ptr : DiPtr;  All : boolean);
{ Dispose DiRec.  This procedure disposes the record pointed to by Ptr.  If
  All is true, it disposes all the records in a the list where Ptr points to
  the first node.
}

procedure Get_MyHelp (Prog : Str_15;  var Name : Str_23);
{ Get my help filename.  This procedure returns the name of a helpfile based
  on the name of the file that the program was started from or on Prog if the
  filename did not contain ".CODE" and was too long.  If a help filename can
  not be built, Name is returned empty.
}

procedure Get_MyFile (Prog : Str_15;  var Name : Str_23);
{ Get my filename.  This procedure returns in Name the complete name of the
  file from which the program named in Prog was started.
}
```

```
implementation
uses  Kernel (IoRsltWd, Alpha,      { System kernel unit }
              E_Rec_P, SysCom, UnitList),
      SysInfo,                      { System information unit }
      Transfer,                     { System file transfer unit }
      ScreenOps,                    { System screen control unit }
      Wild,                         { System pattern matching unit }
      DirInfo,                      { System directory info unit }
      StrOps_U,                     { WDS string ops unit }
      F_Io_U;                       { WDS file I/O unit }

  function NoBreak { (DoWhat : OnOff) : boolean };
  { Uses Kernel unit }
  begin
    with SysCom^ do begin
      if DoWhat = On then  MiscInfo .NoBreak := true
      else if DoWhat = Off then  MiscInfo .NoBreak := false
      else if DoWhat = Toggle then
        MiscInfo .NoBreak := not MiscInfo .NoBreak;

      NoBreak := MiscInfo .NoBreak;
    end { with };
  end { NoBreak };

  function Get_Sc_Width { : integer };
  { Uses ScreenOps unit }
  var T_Port : Sc_Tx_Port;
  begin
    Sc_Use_Port (Sc_Get, T_Port);
    Get_Sc_Width := T_Port .Width + 1;
  end { Get_Sc_Width };

  function Get_Sc_Height { : integer };
  { Uses ScreenOps unit }
  var T_Port : Sc_Tx_Port;
  begin
    Sc_Use_Port (Sc_Get, T_Port);
    Get_Sc_Height := T_Port .Height + 1;
  end { Get_Sc_Height };

  procedure Set_Os_Sc_Size { (H, W : integer) };
  { Uses Kernel and ScreenOps units.  The screen size is maintained by the
    system in the global data sections of both the Kernel and ScreenOps units.
  }
  var T_Info : Sc_Info_Type;  T_Port : Sc_Tx_Port;
  begin
    if (H <> Null) and (W <> Null) then
      begin
        Sc_Use_Info (Sc_Get, T_Info);
        Sc_Use_Port (Sc_Get, T_Port);

        with SysCom^ .CrtInfo do begin
          if H <> Null then  Height := H;

          if W <> Null then  Width := W;

          T_Info .Misc_Info .Width := Width - 1;
          T_Port .Width := Width - 1;

          T_Info .Misc_Info .Height := Height - 1;
          T_Port .Height := Height - 1;

          Sc_ErrorLine := Height - 1;
        end { with };

        Sc_Use_Info (Sc_Give, T_Info);
        Sc_Use_Port (Sc_Give, T_Port);
    end { if };
  end { Set_Os_Sc_Size };
  procedure Get_Sys_Tad { (var Tad : TadRec) };
  { Uses SysInfo unit }
  var H, M, S, T : integer;
  begin
    Tad := NullTad;
    Si_Get_Date (Tad .D);
    Si_Get_Time (H, M, S, T);
    if S >= 30 then  M := M + 1;
    with Tad do begin
      T .Hour := H;
      T .Min := M;
    end { with };
  end { Get_Sys_Tad };
  procedure Set_Sys_Tad { (Tad : TadRec) };
  { Uses SysInfo unit }
  var H, M : integer;
  begin
    with Tad do begin
      H := T .Hour;
      M := T .Min;
    end { with };
    if Tad .D <> NullTad .D then  Si_Set_Date (Tad .D);
    if Tad .T <> NullTad .T then  Si_Set_Time (H, M, 0, 0);
  end { Set_Sys_Tad };
  procedure Get_Pref_Vol { (var S : Str_7) };
  { Uses SysInfo unit }
  begin
    Si_Get_Pref_Vol (S);
  end { Get_Pref_Vol };
  procedure Set_Pref_Vol { (S : Str_7) };
  { Uses SysInfo unit }
  begin
    Si_Set_Pref_Vol (S);
  end { Set_Pref_Vol };
  procedure Get_Sys_Vol { (var S : Str_7) };
  { Uses SysInfo unit }
  begin
    Si_Get_Sys_Vol (S);
  end { Get_Sys_Vol };
  procedure CopyFile { (Src, Dest : Str_23;  var Msg : integer) };
  { Uses Transfer unit }
  begin
    T_Transfer_File (Src, Dest, Msg);
  end { CopyFile };
  function Get_DirInfo { (     Src : Str_255;
                           var Ptr : DiPtr;  var Msg : integer) : boolean };
  { Uses DirInfo unit (which uses Wild unit) and F_Io_U }
  var P, Q : D_ListP;  Pptr : DiPtr;
    procedure Get_Tad (var Ptr : DiPtr);
    { Uses F_Io_U }
    var S : Str_23;  F : FibPtr;  Msg : integer;
    begin
      with Ptr^ do begin
        S := concat ('#000:', Title);
        I_into_S (UnitNum, S, 2, 3);
        if OpenFile (F, S, BlkFile, true, Msg) then
          begin
           Get_File_Tad (F, Tad);
           CloseFile (F, false);
          end { if };
      end { with };
    end { Get_Tad };
    function SetKind (Kind : D_NameType) : FileKind;
    begin
      case Kind of
        D_Vol  : SetKind := FkVol;
        D_Code : SetKind := FkCode;
        D_Text : SetKind := FkText;
        D_Data : SetKind := FkData;
        D_Svol : SetKind := FkSvol;
        D_Dir  : SetKind := FkDir;
      end { case };
    end { SetKind };
  begin { Get_DirInfo }
    Msg := M_NoError;
    new (Ptr);  { Efficient heap usage (so first rec is not at top of heap) }
    case D_DirList (Src, [D_Vol..D_Svol, D_Dir], P, false) of
      D_Okay       : begin
                      Pptr := Ptr;
                       while P <> nil do begin
                         fillchar (Pptr^, sizeof (DiRec), 0);
                         with P^, Pptr^ do begin
                          Kind := SetKind (D_Kind);
                          Blocked := D_IsBlkd;
                          UnitNum := D_Unit;
                          Volume := D_Volume;
                           if D_IsBlkd then
                             begin
                               if D_Kind = D_Vol then  Start := D_NumFiles
                               else  Start := D_Start;
                              Title := D_Title;
                              Size := D_Length;
                              Tad .D := D_Date;
                              Tad .T := NullTad .T;
                             end { if };
                          Q := D_NextEntry;
                         end { with };
                         dispose (P);
                         P := Q;
                         if P <> nil then
                           begin
                             new (Pptr^ .Next);
                             Pptr := Pptr^ .Next;
                           end { if }
                         else  Pptr^ .Next := nil;
                       end { while };
                      if Ptr^ .Next = nil then  { one entry, get time }
                        if Ptr^ .Blocked then  Get_Tad (Ptr);
                    end { case D_Okay };
      D_NotFound   : Msg := ord (I_NoFile);
      D_Exists     : Msg := ord (I_DupFile);
      D_NameError  : Msg := ord (I_BadTitle);
      D_OffLine    : Msg := ord (I_NoUnit);
      D_Other      : Msg := M_Unknown;
    end { cases };
    if Msg = M_NoError then  Get_DirInfo := true
    else
      begin
        DiDispose (Ptr, true);
        Get_DirInfo := false;
      end { else };
end { Get_DirInfo };
procedure DiDispose { (var Ptr : DiPtr;  All : boolean) };
var P : DiPtr;
begin
  if All then
    begin
      while Ptr <> nil do begin
        P := Ptr^ .Next;
        dispose (Ptr);
        Ptr := P;
      end { while };
    end { if }
  else  dispose (Ptr);
end { DiDispose };
procedure Get_Unit (S : Str_15;  var UnitS : Str_5;  var Blk : integer);
label 2;
var   P : E_Rec_P;  I : integer;  A : Alpha;
begin
  UnitS [0] := chr (0);  { UnitS := ''; }
  I := sizeof (A);
  fillchar (A, sizeof (A), ' ');
  Crunch_Str (S, S);
  CapStr (S);
  if I > length (S) then  I := length (S);
  moveleft (S [1], A, I);
  P := UnitList;
  while P <> nil do begin
    with P^ .EnvSib^ do begin
      if SegName = A then
        begin
          UnitS := '#000:';
          I_into_S (VolInfo^ .SegUnit, UnitS, 2, 3);
          Blk := SegAddr;
          goto 2;  { exit while }
        end { if };
    end { with };
    P := P^ .NextRec;
  end { while };
2:
end { Get_Unit };
function Get_File (S    : Str_7;
                   Blk : integer;  var Name : Str_15) : boolean;
label 2, 3;
var   Msg : integer;  P : DiPtr; Ptr : DiPtr;
begin
  Get_File := false;
  Name [0] := chr (0);  { Name := ''; }
  S [0] := succ (S [0]);
  S [length (S)] := '=';
```

```
    if Get_DirInfo (S, Ptr, Msg) then
      begin
        P := Ptr^ .Next;  { First file if any, skip volume record }
        while P <> nil do begin
          with P^ do begin
            if (Blk >= Start) and (Blk < Start + Size) then  goto 2
            else if Blk < Start then  goto 3;
          end { with };
          P := P^ .Next;
        end { while };
      2:
        if P <> nil then
          begin
            Name := P^ .Title;
            Get_File := true;
          end { if };
      3:
        DiDispose (Ptr, true);
      end { if };
  end { Get_File };
  procedure Get_MyHelp { (Prog : Str_15;  var Name : Str_23) };
  var I : integer;  UnitS : Str_5;
  begin
    Name [0] := chr (0);  { Name := ''; }
    Get_Unit (Prog, UnitS, I);  { in case two volumes have the same name }
    if length (UnitS) > 0 then
      if Get_File (UnitS, I, Name) then
        begin
          I := pos ('.CODE', Name);
          if I = length (Name) - 4 then
            begin
              Name [0] := chr (I - 1);  { deletes ".CODE" }
              Name := concat (UnitS, Name, '.HELP');
            end { if }
          else if length (Name) <= 10 then
            Name := concat (UnitS, Name, '.HELP')
          else if length (Prog) <= 10 then
            Name := concat (UnitS, Prog, '.HELP')
          else  Name [0] := chr (0);  { NAME := ''; }
        end { if };
  end { Get_MyHelp };
  procedure Get_MyFile { (Prog : Str_15;  var Name : Str_23) };
  var Blk : integer;  UnitS : Str_5;
  begin
    Name [0] := chr (0);  { Name := ''; }
    Get_Unit (Prog, UnitS, Blk);  { in case two volumes have the same name }
    if length (UnitS) > 0 then
      if Get_File (UnitS, Blk, Prog) then
        Name := concat (UnitS, Prog);
  end { Get_MyFile };
begin { OpSys_U }
  Vv_OpSys_U := Vc_OpSys_U;
  Ck_Version (Vc_Glbs_U, Vv_Glbs_U, Vs_OpSys_U, Vs_Glbs_U);
  Ck_Version (Vc_StrOps_U, Vv_StrOps_U, Vs_OpSys_U, Vs_StrOps_U);
  Ck_Version (Vc_F_Io_U, Vv_F_Io_U, Vs_OpSys_U, Vs_F_Io_U);
  *** ;
end {$Q- OpSys_U }.
```

New Library Disk Directories

What follows is a listing of the directory of the newly released Apple disk. The disks were developed by the Apple SIG.

APP2U02:

```
L.TEXT...............A short but effective text printer with several options.
L.CODE...............Code file for above.
LINECOUNTR.TEXT......Counts the lines of a textfile.
LINECOUNTR.CODE......Code file for above.
PRIME1.TEXT..........Generates primes by the sieve method
PRIME1.CODE..........Code file for above.
PRIME2.TEXT..........Generates primes via the division method
PRIME2.CODE..........Code file for above.
HEXOUT.TEXT..........Prints the equivalent hex code for each keyboard key
HEXOUT.CODE..........Code file for above.
SHELLMSORT.TEXT......Shell-Metzner sort for a textfile, 80 char lines
SHELLMSORT.CODE......Code file for above.
PERUSE.PG.TEXT.......Peruse a text file a page at a time on your CRT
PERUSE.PG.CODE.......Code file for above.
COMPARE.TEXT.........A differential comparator for TEXT files
COMPARE.CODE.........Code file for above.
DELETE.LF.TEXT.......Delete ASCII linefeeds from a textfile.
DELETE.LF.CODE.......Code file for above.
SCANNER.TEXT.........Scan an entire volume looking for string(s)
SCANNER.CODE.........Code file for above.
COMAPP2U02.TEXT......Comments on programs above
DIRAPP2U02.TEXT......You're reading it
```

APP2U04:

```
BASE.DICT.TEXT.......Base level dictionary for SPELLER, no suffixes
BIGG.DICT.TEXT.......Full dictionary for SPELLER, includes suffixes
COMAPP2U04.TEXT......Comments on programs above
DIRAPP2U04.TEXT......You're reading it
```

APP2U05:

```
CHAREDIT.TEXT........Create or change character sets - use as SYSTEM.CHARSET
CHAREDIT.CODE........Code file for above
DOSCAT.TEXT..........Read catalog of Apple DOS 3.3 disk
DOSCAT.CODE..........Code file for above
DOSUNIT.TEXT.........Unit to read DOS 3.3 disks from Pascal - contains DOSSTUFF
DOSUNIT.CODE.........Code file for above
DOSTRANS.TEXT........Program using DOSSTUFF to transfer text from DOS to Pascal
DOSTRANS.CODE........Code file for above
DOSTR.DOC.TEXT.......Documentation for DOSTRANS and DOSUNIT
COMAPP2U05.TEXT......Comments on programs above
DIRAPP2U05.TEXT......You're reading it
```

APP2I06:

```
PASIN1.TEXT..........Part 1 of the Bart Thomas Guerrrila Guide
PASIN3.TEXT..........Part 3 of the GG
PASIN2.TEXT..........Part 2 of the GG
PAS2E.TEXT...........Special part of the GG for the Apple2e only
PBOOKS.TEXT..........Bart's compiled list of books
GG.DOC.TEXT..........Documentation for the files of the Guerrila Guide
INDEX.TEXT...........Expanded index to Jensen and Wirth PASCAL USER MANUAL
WAP.P.8902.TEXT......Text of Washington Apple Pi Pascal column 02/89
WAP.P.8904.TEXT......Text of Washington Apple Pi Pascal column 04/89
WAP.P.8905.TEXT......Text of Washington Apple Pi Pascal column 05/89
R.TRANWARP.TEXT......Review of the Applied Engineering Transwarp card
COMAPP2I06.TEXT......Comments on the above files
DIRAPP2I06.TEXT......You're reading it
```

NOTE: There are no program files or code files on an Information Disk. The files all contain text of some sort, and can be printed out on your printer, or scanned with the Editor. In general, all new SIG related articles, bulletins, NewsLetter articles etc. will go on these disks. Also, some older files which appear on the Library disks will move here if they are still interesting or germane.

USUS

NewsLetter

July/Aug 1989

Copyright 1989, USUS, Inc.
All Rights Reserved
William D. Smith, Editor

Volume 3
Number 6

NewsLetter Publication Dates

NewsLetter	Due date Code/Forms	Due date Articles	Due date Short stuff
November 89	10/01/89	10/13/89	10/20/89
December 89	11/01/89	11/10/89	11/17/89
Jan/Feb 90	01/01/90	01/08/90	01/15/90
Mar/Apr 90	03/05/90	03/12/90	03/19/90

Next NewsLetter coming Sept/Oct

USUS
P.O BOX 1148
LA JOLLA, CA 92038

ADDRESS CORRECTION REQUESTED

[redacted]

A