April 1989

USUS NewsLetter

All the News that Fits, We Print

William D. Smith, Editor

Volume 3
Number 4

From the Editor

by William D. Smith

It seems like I get the NewsLetter out later and later each issue. This month, I was delayed by my taxes (I always wait until the last minute) and a week of very hot weather in southern California (to hot to turn on my Mac in the house). Here it is Friday night, I've been up close to thirty hours and the NewsLetter is finally finished. The next NewsLetter is a two month issue and you should receive it about the end of May. Due to time constraints, I didn't cover my string ops unit as well as I would have liked. If you have questions, comments or improvements, write. Next issue, I will be cover my operating system interface unit.

Q & / | A? ...

*I received the following letter from **David Oshiro** (Nichandros Companies, 2901 Glascock St., Oakland, CA 94601; phone: (415) 533-6000; Fax: (415) 533-6001):*

We are looking in to the idea of hooking up an automatic switching unit (ASU) to our Stride 460 and connecting our Okidata ML 84 and a Diablo 230 to the ASU. We would like to know if you know of any way of sending a string command to the ASU to let it know which printer we want to use? Can this be done while in certain programs? Word7? Flexware?

*The following is a continuation of the thread in the March NewsLetter (page 2). **Chris Jewell** (CIS: 72415, 1757) replied to **Tom Cattrall** (CIS: 72767,622):*

I like your approach to Bob's problem of terminating modules, and would have suggested something quite similar if you hadn't done so. However, I would take advantage of Modula 2's left-to-right short-circuit boolean ops to code DoAllInits as a one-liner:

```
RETURN InitA() AND InitB() AND ... ;
```

From Tom to Chris:

My tendency is to have the list of IF statements in DoAllInits so that the programmer can more easily add, delete, and adapt for more complex code required for any of the Inits.

That said, I'd not thought of stringing the BOOLEAN Inits together and your idea got me to thinking that if it can indeed be done with only BOOLEAN functions, the DoAllInits procedure isn't even needed. The mainline code could be:

```
IF InitA() AND InitB() AND InitC() THEN
  GetToWork;
END;

DoAllTerminations;
```

***Robert Spitzer** (CIS: 75226,3643) replies to Tom also:*

As I expected, I have received some useful and informative replies to the initialization question; I hope William Smith publishes this thread in the NewsLetter. It has been quite informative. The approach I had kludged upon deals with a somewhat more complicated situation - i.e. that my own data structures require multi-step initializations, and must be themselves cleaned up carefully. This is slightly different than the module initialization question. For example, opening a file requires ALLOCATING the file information block, then creating some internal semaphores (each allocated, initialized, etc) for locking, and at each step failure must deallocate only what was already created; same for some other data structures that must be allocated, then internal locking semaphores created, internal error "messages" created, and internal dynamically sized data buffers created (for example the main record descriptor for a task that does real-time data acquisition with interrupt synchronization, multi-buffered write to disk, etc...). What I am doing within the creation procedures are as follows

IF ~init (something) THEN
  RETURN abort (1) END;

IF ~init (another) THEN
  RETURN abort (2)
END;

RETURN TRUE

and

"PROCEDURE abort (n : CARDINAL) :
BOOLEAN" does

IF n > 1 THEN dispose x END
IF n > 0 THEN dispose y END
RETURN FALSE

Yeah, overall initialization is simple:

IF NOT initVitalModules THEN
  nothingWillWork
ELSE doMain; shutDown END.

BEGIN (* doMain *)
  initOnlyWhatIsNeeded;
  IF resourceNotAvailable THEN dontUseIt
  ELSE useIt END;
END (* doMain *).

The trick is to initialize something like the following structure:

```
averager     = RECORD
      error : INTEGER;
      msg   : message;

(* created/loaded/sent dynamically *)

      buffer       : ADDRESS;

(* dynamically variable size *)

      f      : file;

(* opened/changed dynamically; f
contains dynamically created "s :
semaphore" used to control multitask
access to file *)

      ADinfo : ADdescriptor;

(* with a dynamically created interrupt
servicing task/process *)

      dataw,
      msgw    : window;

(* dynamically created, linked to
dynamically redirectable output device,
including bit-map OR on-line-
configure-able serial port or terminal,
each with locking semaphores *)

      variousParams;
      andSoOn

END;
```

and the structures vary as the user does different data collection or analysis operations. I must say, the system is working fairly well now!

The Prez Sez

by Alex Kleider

We are now approaching a year since the last USUS meeting which was held at Tahoe in conjunction with Stride Faire '88 last June. A rejuvenation of USUS seemed to begin at that meeting. Since then the NewsLetter thanks to **William D. Smith** has been published on a regular basis and more recently it has begun to carry more technical material which is what most of the membership wants. **Frank Lawyer** has been remarkably energetic with the Apple Special Interest Group and has results to show for it in the form of a high membership renewal rate within that group. **Hays Busch** continues to keep the organization together as its administrator. (All members should be aware of the fact that without Hays in recent years, it's unlikely that USUS would have survived.) **Bob Clark** continues to serve as Treasurer (a thankless job I am sure) inspite of his own personal obligations. **Sam'l Bassett** has been Chairman of the Board of Directors now for several years and together with other Board Members has been giving much of his/their energies to serving the organization behind the scenes.

But this is just a hand full of people and the membership (judging by renewal rates and comments received) expects more than is being delivered. What do they/you expect and what is it worth to you? Many feel that they should get $25.00 worth of services for their yearly membership fee but I don't think it's realistic to think that way. That money covers the cost of putting out a NewsLetter without much left over and contributes nothing towards material to put into the NewsLetter. The contents of the NewsLetter have to come from the membership. It is USUS's forum for inter member

communication. I join the Editor in asking each member to offer something for the NewsLetter. Let the rest of the membership know what you are doing, how you are doing it and share some code that might have general appeal. It's hard for me to believe that there are many members who truly have nothing to contribute.

The message being presented is "Spend less time asking what USUS should do for you; ask instead what you might be able to contribute to your fellow USUS members."

Apart from "sharing" experiences & code via the NewsLetter, there are three other areas that need attention.

We very much need someone to head up an IBM (and compatibles) SIG to try to do what Frank has been doing in the Apple domain. We already have TI (**Ken Hamai**) and Sage/Stride (**Ray Weglein, Jr**) SIG chairmen but I'm sure extra help would be welcome in all these fields if it were to be offered.

Membership recruitment is an area that has been discussed. Personally I'm not sure how such a project could be implemented but perhaps there's someone amongst the membership that has some ideas in this regard, and better still, someone who would take it on as a project.

The third of these brings us full circle to the topic of an "annual" meeting. Ideally the next one should be in the East but anywhere would undoubtedly be better than none at all. To accomplish this we need a Meeting Chairman and a Local Arrangements Chairman. These might be combined into one person depending on circumstances. Volunteers are very much needed but even if you can't offer you services perhaps you can offer input as to where and when you think the meeting might be held and perhaps you have someone in mind to volunteer!

In closing I'd like to emphasize that this is your (the membership) organization and you collectively will determine if USUS will continue to be around to serve as a forum for those of us with interests that involve the UCSD p-System (and its descendent, the Power System), Pascal, Modula-2 and related topics. Remember that if we/you don't support USUS by renewing membership, USUS will cease to exist and with it will disappear the only resource for this particular computing milieux. Renew and "JOIN" in the activity. Help continue the revival of your organization.

Comments and suggestions from any and all members are welcomed.

Alex Kleider, 1651 Stone Pine Lane, Menlo Park, CA 94025; (415) 327-7916

Administrator Says
by Hays Busch

Elsewhere in this issue is a bit of programming and some comments by yours truly. And after submitting it to the Editor, I began to think about "contributing to USUS" and got a bit angry!

Every one of you reading this has had either a "new member" letter or a "renewal" letter from me in the past 12 months. Both of these have a plea for a "contribution of time or information" that USUS can pass on to the total membership to make the organization more interesting and informative for everyone involved. And damn it, USUS isn't getting much response to this plea!

I'm not complaining for myself. I like to do what I'm doing for the group. But hey guys, I'm only one person! And it takes more than me (and a few others) to keep USUS interesting, alive and well. (It also takes more input from members who have accepted BofD, Officer and SIG responsibility, but that's another story so I'll not go into it now.)

The point is, USUS must have input from the members at large, or ultimately it will fail. It is a small organization now and there is no way it can afford to pay writers to write stuff for the NewsLetter. You members have to do it for "free". Same goes for the BofD, Officers and SIG chairs. If these members don't contribute time and effort, you get the impression that "nothing is happening". The whole thing becomes a CACHE 22. YOU sit back and wait for something to "happen" and the folks who are trying to make it "happen" get discouraged because they don't hear from you! Its like a dog chasing its own tail. He's not getting very far!

I've heard the story, "... but the stuff I have is not 'important', 'good', 'of general interest', (or you fill in the 'excuse'), enough to send in."

Possibly true, but even so, something is better than nothing!

The little programs I have submitted (this one, and the ones in the past) are no great shakes. (Some of you out there can do a lot better!) But even my simple stuff may be of interest to some USUS member. So I send it in.

I can't for the life of me figure out why we still do not have a working IBM SIG in USUS! I can't for the life of me figure out why we can't get owners of Apple and Sage/Stride/Pinnacle to volunteer time to check out and update a SWLibrary disk or disks for their machines!

I do know this. If you are going to help USUS, you need to do it on a "project" or "continuing" basis. If it's a "project", like reviewing a SWLibe volume to see how suitable it is for a certain machine, you need to get it done in a reasonable time...a month? If it is on a "continuing" basis, as a member of the BofD, Officer, Staff person, or SIG leader you need to think in terms of 8 to 16 hours of USUS effort (minimum) each month. If USUS can get 20 to 40 of you doing something like that for USUS each month, there'l be no stopping the group!

My "thanks" to those who are doing it now. For the rest of you...HOW ABOUT IT????

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

Bank Balance	$7,513.30	01-31-89
Income - February 1989		
Dues:		(new/renew)
Student	0.00	0/0
General	500.00	4/10
Professional	200.00	2/0
Institutional	0.00	0/0
Other Income:		
Library fees	0.00	
PowerTools	194.80	
Total Income:	$894.80	
Expenses - February 1989		
Administrator:		
CIS	47.06	
Telephone	16.83	
Photocopies	8.90	
Postage	264.46	
Printing	359.42	
Supplies	88.09	
Other:		
Mail from La Jolla	5.45	
Apple Sig	103.87	
Bank charge	1.00	
Total Expenses	$895.08	
Bank Balance	$7,513.02	02-28-89

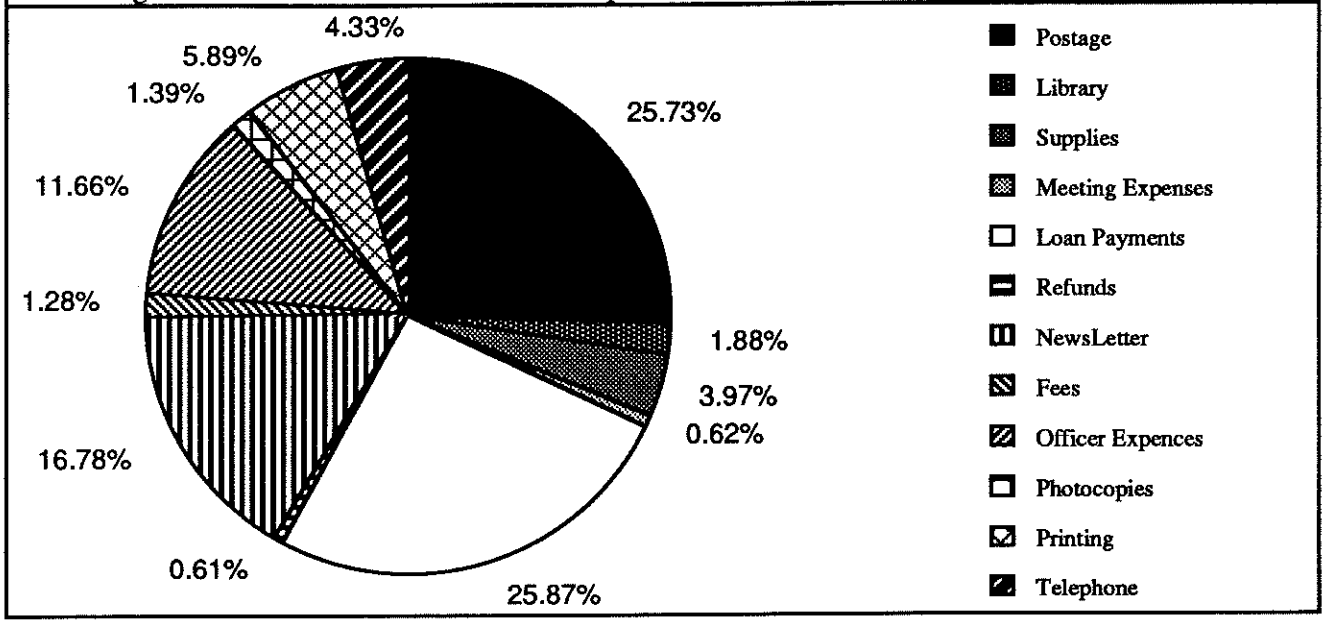
Treasurer's Report (Mar 1989)
by Robert E. Clark, Treasurer

Bank Balance	$7,513.02	02-28-89
Income - March 1989		
Dues:		(new/renew)
Student	50.00	2/0
General	210.00	4/2
Professional	0.00	0/0
Institutional	0.00	0/0
Other Income:		
Library fees	35.00	
PowerTools	206.90	
Misc	5.00	
Total Income:	$506.90	
Expenses - March 1989		
Administrator:		
CIS	41.13	
Telephone	40.48	
Photocopies	4.35	
Postage	212.88	
Printing	304.05	
Other:		
Mail from La Jolla	11.06	
La Jolla Box	36.00	
Other Postage	52.35	
Deposit on NL	500.00	
Officer Reimb.	453.47	
Refunds	44.95	
Bank charge	1.00	
Total Expenses	$1,701.72	
Bank Balance	$6,318.20	03-31-89

1988 Expenditure breakdown

by Robert E. Clark, Treasurer

1988 budget breakdown. Note that the loan is paid off and we will have no loan payments this year.



Apple SIG Doings

by Frank Lawyer

NEW MEMBERS...

Welcome to new Apple SIG members **Jim Stringer** and **Willard Pyles**. Looks like you both have Apple3 machines. Also, **Robert Pigeon**, who has an Apple2. By now you should have all your new member kits, including the Welcome disks, so let us know what else we can do to help you.

PRAISE...

Robert Dell has contributed several floppies full of material for the program library, and for NewsLetter articles. I'm reviewing the material now, and it should show up on library disks in a little while.

FORMATS...

In order to encourage porting of available source code, we can provide any of the Apple Library volumes (archive or Apple-specific) in various formats. The standard format is, of course, UCSD p-System, interchangeable for either the Apple Pascal or Pecan Power System. We can also provide the source code in text file format for Apple ProDOS or DOS 3.3, Apple SOS for the Apple3, or even CP/M text file format. FOTO type files will also translate, but NOT code files. I have changed the disk order form file on the Apple welcome disks to show this. There will be no charge for this service, but you must specify the format you want on the order form.

If anyone decides to port to other compilers, please let me know and we can mention it here to see if there is interest by other folks. Given a reasonable interest level, it might be possible to support another format with a full library conversion.

GS TRANSWARP...

A couple of columns back we mentioned that Applied Engineering had announced a TransWarp GS board, to accelerate the Apple2GS. Based on the recent ads, they are now shipping this board. It runs at 7 MHz instead of the normal 2GS speed of 2.8 MHz. That puts it on a par with a Mac. The company also promises that you will be able to exchange for faster CPUs when they become available in the future. It works either in slot 3 or 4, and you can run at the normal 2GS speed again by going to the control panel. List price on this is $399, and Preferred Computing in Dallas is selling them mail order for $319. However, as of my last check, they had no stock, and so all orders were in a backorder status. Those of you

who are more economy minded may want to wait a few months until the initial demand is satisfied and bigger discounts are available.

TRENTON FAIR...

The date for the Trenton Computer Festival is the weekend of April 22-23. Hopefully, this NewsLetter will get to people by that time *(Sorry, didn't make it)*. For those of you who live in the Trenton, NJ general area, this is very worthwhile to go to. The flea market area covers several acres and usually has loads of bargains. It is just outside Trenton at Trenton State college. This is, as far as I know, the oldest, and one of the largest shows devoted strictly to personal computing. In addition to the outside flea market, there is extensive indoor space, programs and lectures. You can get additional information at (201)-549-7538 or via modem at 215-387-4635.

Harry Baya and I had planned to have a USUS space in the parking lot, sell fleas and get USUS members, but all the spaces were filled, so we will have to forego the booth, but we are going to go, roam the parking lot and have fun. Next year I'll plan further ahead. If you need specific driving directions, or want to know what watering hole we will frequent, call me, my number is at the bottom of the column.

WELCOME DISKS...

The latest welcome disk version is 04/01/89. As always, you can swap for the latest by sending me 2 5.25 floppies, or your old welcome disks. The manuals and software files have been expanded for this version.

LIBRARY DISKS...

Library volume APP2U04 is released as of 03/01/89. It contains two specialized dictionaries for the SPELLER program on volume APP2U03. With a little luck, you'll find that the Editor has stashed a listing somewhere else in this NewsLetter.

The SPELLER program works very well, and the price is low, since it comes from one of our Apple library volumes. I usually work in the p-System Editor, to write letters and such, and until I started to use SPELLER, I didn't have a way to stay within the p-System and check my spelling. In using SPELLER, I found that a lot of words I use regularly, words like "USUS" and "p-System", were, of course, not in the dictionary that was supplied by Charles Rockwell, who wrote it. So, I kept on adding words and adding words. It is very tempting to add just every word you use in your documents, that isn't already in the dictionary, and I initially yielded to the temptation. Soon I had a very large dictionary that was well on its way to filling up one side of a 5.25 inch floppy. Why, just about everything has all those wonderful suffixes. Widget, widgets, widgetness, widgeting, widgetized !

The fact that SPELLER uses "literal dictionary" in a regular TEXT file format is both a good and bad feature. It makes it easy to use ASE to remove errant words, and to print the dictionary at any time, but it does tend to use a lot of space. I finally purified my expanded dictionary, and decided to keep one "simple" dictionary that only holds the root words, no suffixes, no jargon, no technical words that are not part of the "normal" vocabulary. and another which "goes for broke", includes highly technical terms like "compiler", and selected suffixes. Those are files BASE.DICT.TEXT and BIGG.DICT.TEXT on volume APP2U04. I will be expanding these as time goes on.

If you don't have SPELLER, you might want to order it, and if you do have it, you may find some use for the special dictionaries. The documentation is good, and I have added a "hints" section to tell you what I have found in using it.

The Apple specific library volumes currently available are:

Information volumes:
APP2I01, APP2I02, APP2I03, APP2I04

Utilities:
APP2U01, APP2U02, APP2U03, APP2U04

Games:
APP2G01, APP2G03, APP2G04

Print out the order form from your welcome disk. If you have an older welcome disk, some of the volumes may not be listed, so write in the volumes you need. The old "archive" volumes 1, 2, and 9 have all been converted to Apple specific

disks, so those volumes are no longer available. Remember, volumes must be ordered in pairs, and will be supplied on 5.25 inch "flippies". Volumes APP2G03 and APP2G04 compose the Adventure game (350 point version), and MUST be ordered as a pair.

Volunteers to help with the conversion process are gleefully accepted.

BOOKS...

In the February column, we had reviews of three of Tom Swan's books. Tom tells me that the Business Book is no longer available.

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

Mac SIG Doings
by Frank Lawyer

BULLETIN...

I sent a Mac bulletin in mid-February to all USUS members with Mac interests. If you haven't yet told me you have a Mac, and you want to be on the list for periodic bulletins, then drop me a line and request the bulletin. Send me a blank disk and request a copy of the Mac welcome disk. The bulletin expanded on a few points from the last column, and announced that I would soon send a survey, so I can figure out what the best direction is for the SIG, based on how most of you use your machines.

WELCOME DISKS...

I am working on a standard p-System format welcome disk, and expect to be done by the time you read this, so if you would rather have that than the MacWrite format disk, send me a disk, but don't forget to mention that you want p-System format, otherwise you'll get the standard MacWrite disk.

There is a revised Mac welcome disk date 04/01/89. The changes consist mostly of revisions to eliminate the Apple oriented wording and move to Mac oriented wording, and to add a few files. The order form has changed due to adding two Mac specific disks (see below) and the latest columns from this NewsLetter have been added. As always, you can swap your old Mac welcome disk for the new one, just send it to me.

LIBRARY DISKS...

We are announcing two Mac specific library volumes ready as of 04/01/89. These are APPMI01 and APPMI02. With any luck, the Editor has tucked the listings away in some dusty corner of this NewsLetter *(page 23)*. For those not familiar with the naming structure, here is a brief review. "APP" is for Apple, which includes ALL the Apple machine specific volumes. "M" is for MacIntosh. The next letter is "I" for an Information disk, "U" for utilities, "G" for games, "P" for graphics etc. Finally, the last two digits are simply the volume number within the series. So, APPMI02 is the 2nd Information volume for the Apple Macintosh. The structure permits expansion, and has some logic to it, that's why I picked it instead of the time honored system of just numbering the volumes consecutively.

Notice the use of the word "volume" instead of "disk". Because disk capacities vary from computer to computer, the p-System VOLUME is different sizes for different machines. On the Apple2, the p-System volumes are 280 blocks (a block is 512 bytes), while on the Mac, we can have up to 770 blocks in a p-System volume. I am going to standardize on 560 blocks for the Mac, since it is twice 280, and that will keep the pricing uniform. Looking ahead, I can see a possibility that in some cases a Mac volume may have to be a bit larger in order to keep related files together, but I'll cope with that when it comes.

The Mac disks will be 800K HFS format. If you require the old 400K MFS format for some reason, please stipulate that on the order form. It will naturally require twice as many 400K disks to fill an order. An 800K disk can hold up to 4 volumes, and a 400K disk up to 2 volumes. The pricing is $7 per 3.5 inch disk, which includes one volume. Add $2 for each additional volume up to the maximum. So, if you ordered an 800K disk containing 4 volumes (the maximum), you would pay $7 for the basic volume plus $2 each for 3 additional volumes, for a total of $13.

The welcome disk is set up to always have the latest order form and library disk listings. The idea is to print your own order form from the disk, that way it will be specific for the Mac. If you have an older welcome disk that doesn't have

the latest volumes listed, just fill in the names and we'll figure it out. Or, if you want to write, I can send you a copy of the latest order form.

It is important that you mail your ORDERS to PO Box 1148, La Jolla CA 92038. Sending them to me will only delay things. I will be glad to handle requests for order forms, new welcome disks etc.

Please note that these first two volumes are Information volumes as contrasted with utilities or games. There are no program files or code files on an Information volume. 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 Apple and Mac SIG related articles, bulletins, NewsLetter articles etc. will go on these volumes. Also, some older text files which appear on the Library disks will move here if they are still interesting or germane.

I have started converting programs also. These disks will be along shortly, as time permits. We will continue the idea of supplying program volumes with both .TEXT and .CODE files. I will probably use Pecan version IV.2.2 for the code files, since that is that latest Mac version. Those with earlier versions can simply compile the .TEXT files. Again, if you are interested in helping in this effort, let me know. This will result in Mac-specific programs that will run on the p-System on a Mac Plus and up. I have access to an older Mac 512K with the old ROMs, and I can do a limited amount of regression testing on it. First I have to find out whether IV.2.2 runs on it or whether it will be IV.1.3.

One word on the USUS "archive" volumes. These volumes will be converted to Mac format, because I believe they should be available in all machine formats that we presently support, if only for the purists, and to serve as a base for further efforts. However, this is not a high priority item on my wish list. It takes about 2 hours per volume to convert from Apple2e format to Mac format, so I will do it on a "as needed" basis. That means that if you should order Mac format old volumes 1, 2, 3, 4 for instance, I may have to convert some of these just for your order. In the future, some of these volumes may be withdrawn as the programs are converted into Mac-specific programs.

LETTERS...

I got a short letter from **David Babb** who said that Santa brought him a new Mac Plus, and that he is learning to use it. Also, a short note from **James Harding**, who promises a longer letter later. I'm waiting!...Two letters from **David Craig**, who has also sent along some Pascal stuff which should run on the Mac and Lisa both. I'm working on the programs, and they will move to library disks later... A short EasyPlex from **Matt Snyder** who wants to be able to interchange code between his Mac and his Apple2. I'll cover the questions and answers next time.

INTERESTING PRODUCTS...

A while ago, I bought a product called "Two in a Mac". I took a look at the documentation, which was much more brief than I would have liked, and decided that it was not quite what I had hoped it would be. Then it gathered dust for a long time. That seems to have been my error.

In February, after writing the Mac SIG column for the NewsLetter, I decided to do a little homework on how I was going to move files between the Apple2e and the Mac Plus. Of course, there are several ways to do this, and I may have an article on this later. Anyway, because my machines are quite a distance apart, file transfer via cable was out. So I decided to break out my "Two in a Mac" (TIM) and see what it would do for me. The product is an Apple2e emulator which runs on the Mac, and allows almost all 2e programs to run. My immediate interest was in file transfer, so that was what I checked first.

After you boot the program, you get a graphics screen on the Mac which represents the facilities you might normally have on your 2e. Four disk drives, a modem, and a printer are all there, and they function like your Apple2e setup. You can trade two 5.25 inch pseudo-drives for one 3.5 inch microfloppy drive that corresponds to your external 3.5 inch Mac drive. That's good, because after all, it's tough to get a regular floppy into the 3.5 inch slot. Now for the fun. You can mount a 3.5 inch ProDOS volume in the external Mac drive, boot ProDOS from the pseudo 5.25 disk (supplied with TIM), and presto you're running ProDOS on the Mac. Now this setup

will not set any modern CPU speed records, because despite the power of the 68000, the emulation process eats a lot of cycles. However, everything works, and ProDOS (version 1.2) does everything it does on the Apple2e.

Through a supplied conversion program, TIM lets you convert files back and forth between ProDOS (or DOS 3.3, for the ancients among you!) and Mac ASCII text files. You can then import those into MacWrite, or use the conversion routine that comes with the Mac p-System to get them into a more useful format for the Mac.

Wait a minute! If we started with p-System .TEXT files, how did we get the ProDOS files? Why we used Universal File Conversion (UFC) from Quality Software. This product was reviewed some months ago in the Apple SIG column. It will convert files among Pascal, SOS, DOS3.3, ProDOS, and even CP/M. One of its best features is that you can use wildcard specifications (the infamous =.TEXT, for instance) to do a whole disk, then just come back later, and all the files have been converted to a different format. Actually, UFC is quite fast, and will translate an Apple Pascal volume to whatever in about a minute.

IN THE FUTURE...

In future columns I will have a more detailed review of the "Two in a Mac" program, as well as the Pecan p-System versions IV.1.3 and IV.2.2, and then, depending on feedback I get from you, some other language implementations. I hope to get the survey out to you soon, so I can tell who is using what packages. We will probably only be able to treat the most popular ones. Of course, anyone who wishes to write about XYZ Modula is welcome to do so. Send me your articles, notes, incantations and whatevers. My address is, as usual, at the end of the column.

LATE BREAKING APRIL NEWS...

In a departure from their normal policy which dictates January as the month of major product announcements, Apple Computer held a sparsely attended seminar on April first, at the Fargo ND Red Roof Inn. They announced a new version of the popular Macintosh, this model to be designated the MacAF. Sporting a hitherto unannounced 680000 processor running at a brisk 67MHz, and using a 64 bit address and data bus, data transfer rates are said to exceed the capabilities of current measuring instruments. The MacAF can address over 2048 gigabytes using the NIY 64 megabyte SIMMs available as plug-in modules through Apple dealers.

The machine sported a vertical screen which could be divided into 4 horizontal screens of normal Mac aspect ratio. This was to allow the user to view the 4 separate Mac "sessions" allowed by the multi-user proprietary operating system. Using the new system default 5 pt Monaco font, up to 16 output screens can be active at one time. It was said the machine could run MS-DOS, CP/M and UNIX as well as OS9 and the Mac MultiFinder in addition to the p-System, which was the native OS...Asked why APRIL was chosen for such an important announcement, the Apple spokesperson replied that any FOOL knows that its the nicest month in Fargo. Asked "What's next?" The reply was "a small black cube about a foot on a side."

Although it lacks the normal floppy drives, the MacAF comes equipped with an high capacity internal WORS (Write Once Read Sometimes) drive containing the collected works of Donald Knuth, the Library of Congress, and a USUS welcome disk. A cassette output port to allow user backups may be ordered as an extra cost option.

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

TI SIG News

by Ken Hamai

Hi there! I want to welcome back all of you members who have renewed and also welcome in all of you new members who just signed up. Hey, did you all see the big write-up that **Jim Horn** (CompuServe TI SysOp) gave us in the Computer Shopper! If you didn't, go down to your local library and check out the January 1989 issue, page 470!

One of the most asked questions regarding the use of the Pascal system is: "Will it work with MY floppy disk drives?" Typically, the person asking the question has a system that has been

upgraded from the standard TI supplied configuration which consisted of up to three single sided single density disk drives (this is assuming you are using the P-Box expansion, not the boxcars). This gave you 360 sectors or 180 Pascal blocks of disk space. Remember that a sector is 256 bytes and a Pascal block is 512 bytes (two sectors). All of the Pascal software supplied by TI came in this 180 block format

The disk controller card that TI supplied for the system is actually capable of controlling up to three double sided disk drives using single density format. Therefore if you have double sided disk drives, your Pascal system can use this extra capacity for 360 block disks.

Third party disk controllers from Myarc and Corcomp are capable of giving you double sided double density disks. If you have one of these controllers and the double sided double density disk drives, you already know that you can get 1440 sectors. This is 720 blocks in the Pascal system.

There is a slight problem with using the Myarc disk controller at 720 blocks, in that the disk controller must be informed that it has a 720 block disk. Users have gotten around this problem in a couple of different ways, a topic which will be covered in the next installment. In any case, if you use the Myarc default format, you will get 16 sectors per track which will gives you 640 blocks.

The third party controllers are also capable of controlling four disk drives. But the TI Pascal software was only set up for three drives, volumes 4, 5, and 9. Where a fourth disk drive is used, one will need to tell the system to access this fourth drive. This will also be covered in the next installment.

One more thing for those with 80 track capability, the Pascal system will handle those too, so if you go all out, you can run the 720k, 3.5 inch disk drives for a total of 1440 blocks per disk.

If you stayed with the TI single side single density system, probably 90% of you out there, you probably did not even know about the giant bug in the DFORMAT program. This program is supposed to format your disk for whatever amount of blocks/density/tracks you specified in the input section of the program. Well, no matter what you put in above single sided single density, 40 tracks, you only got single side single density, 40 track disk format.

The trick here is to format all of your disks using the TI operating system and your favorite disk manager program, then zero the disks with the Pascal Filer to the number of blocks you have available. I use DM1000 to format and verify my disks for DSDD (1440 sectors) and then zero my disks for 720 blocks.

At this time, I don't know if the Pascal system will work with the new Myarc hard disk controller and a hard disk. Will get back to you as soon as I find out! Frank! Got your ears on?

--Questions and Answers--

Member **Jim Fetzner**, USAF, Germany, writes: Do any of the other UCSD language compilers run on the 99/4a? Well, Jim, as far as I know, no others were ported over to the TI. This situation may change, depending on the fate of the Geneve 9640 Pascal runtime.

--Tricks and Tips--

Member **Andrew Becker**, NY writes: He has a ton of USUS library programs in IBM 640 block format and needs to convert it to TI-99 format. He has a Xerox 1810 and a TI. Well Andy, if you also have the Pascal system for your Xerox, you ought to be able to just hard wire both your Xerox serial port and the TI serial port together and use RemTalk to send the stuff over.

By the way, Andy also figured out a way to use his VT-52 compatible terminal with the TI p-System. He says the great thing is that he now has an 80 column screen! Not so great is the screen update time, which can take up to 16 seconds to write a new screen. Andy, hope you got your baud rate set up to the maximum 9600!

--Library Volunteers--

Ed Livingston, NC writes: He would be interested in helping with work of Mr. Carl Schuneman, in applying the library to the 4a. Well, Ed, if you get nothing back from Carl, let me (Ken Hamai) know what library disk you want to work on and what disk format you use with your p-System. Then keep an eye on your

mail for a disk envelope! The same goes for anyone else out there who wants to help.

Till next time..Try it! It works for me!

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

UCSD text file to Turbo text file Filter

By Hays Busch

I have been experimenting with USUS PowerTools to find out how it works transferring files between my SAGE II (upstairs) over the Hayes Smartmodem 1200 to the IBM AT (downstairs). And it works just great!

In the process, I transferred a text file written with the ASE Editor and UCSD Pascal, Version IV.2.2. to the IBM. So, I decided to see how it would compile under TURBO Pascal, Version 4.0. I called it up and lo and behold, lots of garbage characters and no format (indentation) in the file. Knowing that would not compile, I had the choice of doing a hand editing job or writing the following "filter" program.

Unfortunately, once the program is converted, if you transfer it back to the SAGE II and try to edit it with ASE, the result is a total disaster! So some other member might want to write a program ("RECONVERT" ?) to get TURBO Pascal files into the UCSD Editor's format. But I'd had enough fun by this time and did not attempt it!

Should not be too hard to do. HINT: Use "PATCH" to find out what is in the first 1024 bytes of the UCSD Editor's file *(for new text files, two blocks of chr (0))*. You will have to provide this before you start the text portion of the file.

Final note. If you DO NOT convert the file with this program, PowerTools will send it back to the SAGE II and ASE will find it perfectly acceptable and in the exact format it wants. PowerTools is a "gem" when it comes to transferring files back and forth between machines.

```
{ A program written in Version 4.0 of
Turbo Pascal.  It removes the imbedded
format commands used by the UCSD Pascal
Screen Editor and reformats the file to
its original indentation.  After
conversion, the file is fully
compatible with the Turbo Pascal Screen
Editor.  This program may be used by
USUS Members for any purpose.  Written
March, 1989 by A. H. Busch, 2193
Montane Drive East, Golden, CO. 80401 }

program Convert;

{ Convert UCSD Editor Text Files to
Turbo Editor Text Files }

const      TempName    = 'TEMP.@@@';

var  FileName    : string [16];
InFile      : text;
Outfile     : text;

  procedure ReFormatFile;

{ Calculates spaces required to restore
original indentation and writes them to
the file. }

  var       Ch    : char;
Index : integer;
Spaces      : integer;

  begin
    read (InFile, Ch);
    { Strip eighth bit }
    Spaces := ord (Ch) mod 128 - 32;

    for Index := 1 to Spaces do
      write (Outfile, ' ');
  end { ReFormatFile };

  procedure ConvertFile;

  var       Ch    : char;
Index : integer;

  begin
    assign (InFile, FileName);
    reset (InFile);

    assign (OutFile, TempName);
    rewrite (Outfile);

    { Read past the text file header }
    for Index := 1 to 1024 do
      read (InFile, Ch);

    while not eof (InFile) do begin
      read (InFile, Ch);

      { Strip eighth bit }
      Ch := chr (ord (Ch) mod 128);
```

```
      { DLE preceeds format character.}
      if Ch = chr (16{DLE}) then
        ReFormatFile
      else if Ch = chr (13{EOL}) then
        { Write <CR><LF> to file.}
        writeln (OutFile)
      else if Ch >= ' ' then
        { a printable character }
        write (Outfile, Ch);
    end { while };

    close (InFile);
    close (OutFile);
    erase (InFile);
    rename (OutFile, FileName);
  end { ConvertFile };

begin { Convert }
  writeln ('Convert UCSD Editor •••
          •••Text Files to Turbo •••
          •••Editor Text Files');

  writeln;
  write ('Convert what file? ');
  readln (FileName);

  if length (FileName) > 0 then
    ConvertFile;
end { Convert }.
```

The three •'s at the end of a line and the beginning of the next line (in the preceding program) mean that the line must be appended together and the •'s removed for the program to compile.

WDS String Ops Unit

By William D. Smith

During the course of my programing, I have tended to use a minimal set of input / output routines (read a string, write a string and get a command form the keyboard). I use this unit to format the string for output and extract the data value from the string on input. At this time, it only contains routines for the data types which my programs have needed to use for their I/O. You will notice, that it contains no routines for handling real numbers. I have not had to address this issue yet outside of a fixed point money routine. *(A recent article in the "Journal of Pascal, Ada and Modula2" (Jan/Feb 89) dealt with formatted real I/O in a real neat manner)*

All the data conversion routines have three parts. An `_into_S` procedure which places the data in the string in a formatted manner. The `_fr_S` procedure which extracts the value of the data from the string (beginning at the given location). And the `S_to_` procedure which converts the value in the string to the data type.

This unit also contains three procedures `CapStr` (converts a string to upper case), `Crunch_Str` (remove all spaces from a string) and `AppendText` (which appends '.TEXT' to strings consisting of filenames) which modify strings and one function (`Cap`) which returns the uppercase character when passed a lowercase character.

The data types supported are integer, hex, date, time and `YesNo`. `YesNo` can be used as a two, three or four way toggle which is used to let the users turn on or off various options. For instance, given a list of the files to be printed, the user can select `Yes` or `No` and only those files with the `Yes` flag will be printed.

```
{ WDS string conversion unit       [1.11] --- 09 Apr 89 } { |xjm$d|nx|f8|e|. }
{$Q+}
{$C (c)  William D. Smith  -- 1988, 1989,  All rights reserved.              }
{ File:          StrOps_U.Text              Version 1.11    09 Apr 89

  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) 1988 to 1989.

  System:        Power System version IV.2.2

  Compiler:      Power System Pascal Compiler
```

```
  Keywords:     WDS Glbs_U Globals Unit

  Description:  String conversion unit.  This unit contains procedures to convert
                between strings and other common data types.

  Change log:   (most recent first)

Date        Id   Vers  Comment
---------   ---  ----  -------
09 Apr 89   WDS  1.11  Implemented D_fr_S.  Changed S_to_D to use it.
09 Mar 88   WDS  1.10  Move AppenedText here from Tx_Io_U.
29 Oct 87   WDS  1.09  Fixed error in I_into_S and H_into_S.
20 Oct 87   WDS  1.08  Added Vs_StrOps_U and its use.
16 Sep 87   WDS  1.07  Added H_into_S, allowed S_to_I and I_fr_S to input Hex.
31 Aug 87   WDS  1.06  Fixed for IV.22.
16 Jul 87   WDS  1.05  Put in version control.
19 Jun 87   WDS  1.04  Removed space btwn time and am/pm.
12 Jun 87   WDS  1.03  Added YesNo stuff, I_Len, and D_fr_S.
28 May 87   WDS  1.02  Added Crunch_Str, S_to_I, S_to_D.
27 May 87   WDS  1.01  Added T_into_S.
08 May 87   WDS  1.00  Started with I_into_S, D_into_S, I_fr_S.
}

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

unit StrOps_U;

interface  {$ StrOps_U [1.11] 09 Apr 89 }

uses  Glbs_U;

const Vc_StrOps_U = 5;  { 09 Mar 88 }
      Vs_StrOps_U = 'StrOps_U';

var   Vv_StrOps_U : integer;

  function Cap (Ch : char) : char;
  { Captialize.  This function returns the uppercase of Ch if Ch is a lowercase
    character.  Otherwise it just returns Ch.
  }

  procedure CapStr (var S : Str_255);
  { Captialize string.  This procedure changes all lowercase characters in S to
    uppercase.
  }

  procedure Crunch_Str (var Src : Str_255;  var Tgt : Str_255);
  { This procedure copies Src to Tgt removing all blanks.  Src is unchanged. }

  procedure I_into_S (I : integer;  var S : Str_255;  Index, Len : integer);
  { Integer into string.  This procedure places the string representation of the
    integer I into the string S.  Index is the location in S where the field
    starts, Len is the width of the field.  The string representation of I is
    placed into S starting at Index + Len - 1 going backwards.  If the string
    representation of I is > Len, then the field is filled with stars.  If Len
    is Null, the integer is left justified.
  }

  procedure I_fr_S (var I : integer;
                    var S : Str_255;  var Index : integer);
  { Integer from string.  This procedure extracts an integer, starting at Index
    from the string S.  Index is left pointing at the character past the last
    digit in the integer.  If S [Index] is not a valid digit, I is returned as
```

```
    zero and Index is unchanged.  If S [Index] is a "$", the number is input as
    an hexadecimal value.  S is unchanged.
}
function S_to_I (S : Str_7;  var I : integer;
                 Min, Max : integer) : boolean;
{ String to integer.  This function converts the string S to an integer I.  If
    I is between Min and Max (inclusive) then the function returns true.  The
    function returns false if there is an error in the string.  I is undefined
    if S_to_I returns false.  If the string starts with a "$", the number is
    input as a hexadecimal value.
}
procedure H_into_S (I : integer;  var S : Str_255;  Index, Len : integer);
{ Hex into string.  This procedure places the string representation of the
    integer I into the string S in hexadecimal format.  Index is the location in
    S where the field starts, Len is the width of the field.  The string
    representation of I is placed into S starting at Index + Len - 1 going
    backwards.  If the string representation if I is > Len, then the field is
    field with stars.
}
procedure D_into_S (Dr : DateRec;  var S : Str_255;  Index : integer);
{ Date into string.  This procedure places the string representation of the
    date Dr into the string S begining at the Index location in S.  Eight spaces
    are used for the date in the form "mm-dd-yy".  If Dr = NullTad .D, nothing
    is done.  No range checking is done (the string must be at least eight
    characters long, Index must be > zero, and Index + 8 must be < length (S)).
}
procedure D_fr_S (var Dr : DateRec;  var S : Str_255;  Index : integer);
{ Date from string.  This procedure extracts a date, starting at Index from
    the string S.  Index points to the character after the last character in the
    date.  The date must be in the form "mm?dd?yy" ("?" is any non-numeric
    character).  The date takes up a maximum of 8 spaces.  " m? d? y" is not
    allowed because there is more then one character (space counts as one
    character) between the digits.  If there is an error, Dr is returned as
    NullTad.D and Index is unchanged.  S is unchanged.
}
function S_to_D (S : Str_9;  var Dr : DateRec) : boolean;
{ String to Date.  This function converts the string S in the form "mm?dd?yy"
    ("?" is any non-numeric character) or "+|-<num>" into the date Dr.  The
    function returns true only if the string contains a valid date.  Dr is
    undefined if S_to_D returns false.  If the first non-blank character in S is
    a "+" or "-", the date is incremented/decremented by the following number.
    If no number, one is assumed.  If Dr is NullTad .D, this latter form is not
    allowed and returns an error.
}
procedure T_into_S (Tr : TimeRec;  AmPm : boolean;
                    var S : Str_255;  Index : integer);
{ Time into string.  This procedure places the string representation of the
    time Tr into the string S begining at the Index location in S.  AmPm
    determines the format ("11:59pm"|"23:59").  Seven|five spaces are used for
    the time.  If Tr = NullTad. T, nothing is done.  No range checking is done
```

```
    (the string must be at least seven|five characters long, Index must be >
    zero, and Index + 7|5 must be < length (S)).
  }
  procedure T_fr_S (var Tr : TimeRec;
                    var S : Str_255;  var Index : integer);
  { Time from string.  This procedure extracts the time from the string S.  The
    time must be in the format "hh?mm" ("?" is any non-numeric character,
    allowed forms are "", ":m", "h", "h:m", etc.  "9: 9" is not allowed because
    there are two characters between hour and minutes).  Index is incremented to
    point to the character past the time.  If there is an error, Tr is returned
    as NullTad .T and Index is unchanged.  S is unchanged.
  }
  function S_to_T (S : Str_7;  var Tr : TimeRec) : boolean;
  { String to time.  This function converts the string in the form "hh?mm" ("?"
    is any non-numeric character) into the time Tr.  The function returns true
    only if the string contains a valid time.  If the function returns false, Tr
    is undefined.
  }
  procedure Yn_into_S (Yn : YesNo;  var S : Str_255;  Index, Len : integer);
  { YesNo into string.  This procedure puts the string representaion of Yn into
    the string S begining at Index.  The width of the field is Len characters
    long (Len must be 3 or greater).  The value in the field is right justified.
  }
  procedure Yn_to_S (Yn : YesNo;  var S : Str_5);
  { YesNo to string.  This procedure converts Yn to the string S. }
  procedure Yn_Fr_S (var Yn : YesNo; var S : Str_255;  var Index : integer);
  { YesNo from string.  This procedure extracts a YesNo, starting at Index from
    the string S.  Index is left pointing at the first space (or end of string)
    past the YesNo.  If S [Index] is not a valid letter (" ", "Y", "N", "O"), Yn
    is returned as Neither and Index is unchanged.  Lowercase characters are Ok
    and only the first letter is checked.  S is unchanged.
  }
  function S_to_Yn (S : Str_5;  var Yn : YesNo;  Allowed : YnSet) : boolean;
  { String to YesNo.  This function returns true if the string S represents one
    of the values for YesNo in the allowed set.  Yn is undefined if S_to_Yn
    returns false.
  }
  procedure AppendText (var S : Str_255;  Caps : boolean);
  { Appends ".Text" to S if S is not empty and does not end with ".TEXT" (may be
    mixed case), ".", or ":".  If S ends with ".", the period is deleted.  If
    Caps is true the string is returned as all captials.
  }
implementation
  function Cap { (Ch : char) : char };
  begin
    if (Ch >= 'a') and (Ch <= 'z') then
      Cap := chr (ord (Ch) - 32)
    else  Cap := Ch;
  end { Cap };
```

```
procedure CapStr { (var S : Str_255) };
var I : integer;
begin
  for I := 1 to length (S) do begin
    if S [I] >= 'a' then
      if S [I] <= 'z' then
        S [I] := chr (ord (S [I]) - 32);
  end { for };
end { CapStr };

procedure Crunch_Str { (var Src : Str_255;  var Tgt : Str_255) };
var I, J : integer;
begin
  J := 0;

  for I := 1 to length (Src) do
    if Src [I] <> ' ' then
      begin
       J := J + 1;
        Tgt [J] := Src [I];
      end { if };

  Src [0] := chr (J);
end { Crunch_Str };

function I_Len (I : integer) : integer;
{ Integer length.  This function returns the number of
  characters needed to display/print the integer I.
}
var J : integer;
begin
  if I < 0 then
    begin
      J := 1;
      I := - I;
    end { if }
  else  J := 0;

  if I < 10 then  I := 1
  else if I < 100 then  I := 2
  else if I < 1000 then  I := 3
  else if I < 10000 then  I := 4
  else  I := 5;

  I_Len := I + J;
end { I_Len };

procedure I_into_S { (I : integer;  var S : Str_255;
                      Index, Len : integer) };
var J : integer;  Neg : boolean;  Ss : Str_7;
begin
  if Len = Null then  Len := I_Len (I);

  J := Index + Len - 1;

  if I = - maxint - 1 then
    begin
      Ss := '-32768';
      J := J - length (Ss);
      moveleft (Ss [1], S [J + 1], length (Ss));
    end { if }
  else
    begin
      Neg := I < 0;

      if Neg then  I := - I;

      repeat
```

```
        S [J] := chr (ord ('0') + (I mod 10));
        I := I div 10;
        J := J - 1;
      until (I = 0) or (J < Index);
      if (I > 0) or ((J < Index) and Neg) then
        fillchar (S [Index], Len, '*')
      else if Neg then  S [J] := '-';
    end { else };
end { I_into_S };

procedure I_fr_S { (var I : integer;
                    S : Str_255;  var Index : integer) };
var Ch : char;  Hex, Neg, NotDone : boolean;  Size : integer;
begin
  I := 0;
  if Index > 0 then
    if Index <= length (S) then
      begin
        Hex := false;
        Neg := false;
        NotDone := true;
        Size := 10;
        if S [Index] = '$' then
          begin
            Hex := true;
            Size := 16;
          end { if }
        else if S [Index] = '-' then
          begin
            Neg := true;
            Index := Index + 1;
          end { if }
        else if S [Index] = '+' then
          Index := Index + 1;
        while NotDone do begin
          if Index > length (S) then  NotDone := false
          else if S [Index] < '0' then  NotDone := false
          else if S [Index] <= '9' then
              begin
                I := (I * Size) + ord (S [Index]) - ord ('0');
                Index := Index + 1;
              end { if }
          else if Hex then
            begin
              Ch := Cap (S [Index]);
              if Ch >= 'A' then
                if Ch <= 'F' then
                  begin
                    I := (I * 16) + ord (Ch) - ord ('A') + 10;
                    Index := Index + 1;
                  end { if }
                else  NotDone := false
              else  NotDone := false;
            end { else if }
          else  NotDone := false;
        end { while };
        if Neg then  I := - I;
      end { if if };
end { I_fr_S };
```

```
function S_to_I { (S : Str_7;  var I : integer;
                   Min, Max : integer) : boolean };
var J : integer;
begin
  S_to_I := false;
  if length (S) > 0 then  { non-empty }
    begin
      J := 1;
      I_fr_S (I, S, J);
      if J > length (S) then  { no errors }
        if I >= Min then
          if I <= Max then
            S_to_I := true;
    end { if };
end { S_to_I };

procedure H_into_S { (I : integer;  var S : Str_255;
                      Index, Len : integer) };
var J, K : integer;
begin
  J := Index + Len - 1;
  repeat
    K := I mod 16;
    if K > 9 then  S [J] := chr (ord ('A') - 10 + K)
    else  S [J] := chr (ord ('0') + K);
    I := I div 16;
    J := J - 1;
  until (I = 0) or (J < Index);
  if I > 0 then  fillchar (S [Index], Len, '*');
end { I_into_S };

procedure D_into_S { (Dr : DateRec;  var S : Str_255;  Index : integer) };
begin
  if Dr <> NullTad .D then
    begin
      fillchar (S [Index], 8, '0');
      S [Index + 2] := '-';
      S [Index + 5] := '-';
      with Dr do begin
        I_into_S (Month, S, Index,     2);
        I_into_S (Day,   S, Index + 3, 2);
        I_into_S (Year,  S, Index + 6, 2);
      end { with };
    end { if };
end { D_into_S };

procedure D_fr_S { (var Dr : DateRec;  var S : Str_255;
                    var Index : integer) };
var M, D, Y, I, DaysInMonth : integer;
begin
  Dr := NullTad .D;
  I := Index;
  I_fr_S (M, S, I);
  I := I + 1;
  I_fr_S (D, S, I);
  I := I + 1;
  I_fr_S (Y, S, I);
  if (Y >= 0) and (Y <= 99) then
    if (M >= 1) and (M <= 12) then
      if D >= 1 then
```

```
        begin
          if M = 2 then
            if ((Y mod 4) = 0) and (Y mod 400) <> 0) then  DaysInMonth := 29
            else  DaysInMonth := 28
          else if M in [4, 6, 9, 11]  then  DaysInMonth := 30
          else  DaysInMonth := 31;
          if D <= DaysInMonth then
            with Dr do begin
              Month := M;
              Day := D;
              Year := Y;
              Index := I;
            end { if with };
        end { if };
end { D_fr_S };

procedure IncDate (var Dr : DateRec;  I : integer);
{ Increment date.  Increment the date Dr by I days. }
begin
  with Dr do begin
    while I > 0 do begin
      if (Day = 31) or
         (  (Day = 30) and (Month in [4, 6, 9, 11]) ) or
         ((Day = 29) and (Month = 2) ) or
         ( (Day = 28) and (Month = 2) and
           ( ((Year mod 4) <> 0) or ((Year mod 400) = 0) ) then
        begin
          Day := 0;
          if Month = 12 then
            begin    Month := 0;    Year := Year + 1;    end { if };
          Month := Month + 1;
        end { if };
      if Day = 0 then
       if I > 28 then
          begin    Day := Day + 28;    I := I - 28;    end { if }
        else { if I <= 28 then }
          begin    Day := Day + I;    I := 0;    end { else }
      else { if Day > 0 then }
        begin    Day := Day + 1;    I := I - 1;    end { else };
    end { while };
  end { with };
end { IncDate };

procedure DecDate (var Dr : DateRec;  I : integer);
{ Decrement date.  Decrement the date Dr by I days. }
label 1;
begin
  with Dr do begin
    while I > 0 do begin
      if Day = 28 then
        if I > 28 then
          begin    Day := 0;    I := I - 28;    end { if }
        else { if I <= 28 then }
          begin    Day := Day - I;    I := 0;    end { else }
      else { if Day <> 28 then }
        begin    Day := Day - 1;    I := I - 1;    end { else };
      if Day = 0 then
        begin
          Month := Month - 1;
          if Month = 0 then
```

```
            begin
              Year := Year - 1;
              if Year = 0 then  goto 1;  { Dr = NullTad .D }
              Month := 12;
            end { if };
          if Month in [1, 3, 5, 7, 8, 10, 12] then  Day := 31
          else if Month in [4, 6, 9, 11] then  Day := 30
          else { if Month = 2 then }
            if ((Year mod 4) = 0) and ((Year mod 400) <> 0) then
              Day := 29
            else  Day := 28;
        end { if };
    end { while };
  end { with };
1:
end { DecDate };

function S_to_D { (S : Str_9;  var Dr : DateRec) : boolean };
label 1;
var I : integer;  Ok : boolean;
begin
  S_to_D := false;
  delete (S, 1, scan (length (S), <> ' ', S [1]));
  if length (S) > 0 then
    if (S [1] = '+') or (S [1] = '-') then
      begin
        if Dr = NullTad .D then  goto 1;  { error }
        Ok := S [1] = '+';
        if length (S) = 1 then  I := 1
        else
          begin
            delete (S, 1, 1);
            if not S_to_I (S, I, 0, 32767) then  goto 1;
          end { else };
        if Ok then  IncDate (Dr, I)
        else  DecDate (Dr, I);
        S_to_D := true;
      end { else if }
    else
      begin
        I := 1;
        D_fr_S (Dr, S, I);
        if (Dr <> NullTad .D) and (I > length (S)) then  S_to_D := true;;
      end { else };
1:
end { S_to_D };

procedure T_into_S { (Tr : TimeRec;  AmPm : boolean;
                      var S : Str_255;  Index : integer) };
var I : integer;
begin
  if AmPm then  I := 7
  else  I := 5;
  if  Tr = NullTad .T then  fillchar (S [Index], I, ' ')
  else
    begin
      fillchar (S [Index], I, '0');
      S [Index + 2] := ':';
```

```
      if AmPm then  S [Index + 6] := 'm';
      with Tr do begin
        if AmPm then
          begin
            if Hour >= 12 then
              begin    I := Hour - 12;    S [Index + 5] := 'p';    end { if }
            else
              begin    I := Hour;    S [Index + 5] := 'a';    end { else };
            if I = 0 then  I := 12;
          end { if }
        else I := Hour;
        I_into_S (I,   S, Index,     2);
        I_into_S (Min, S, Index + 3, 2);
      end { with };
    end { else };
end { T_into_S };

procedure T_fr_S { (var Tr    : TimeRec;
                    var S     : Str_255;
                    var Index : integer) };
var H, I, M : integer;
begin
  Tr := NullTad .T;
  I := Index;
  I_fr_S (H, S, I);
  I := I + 1;
  I_fr_S (M, S, I);
  if (H >= 0) and (H < 24) then
    if (M >= 0) and (M < 60) then
      with Tr do begin
        Hour := H;
        Min := M;
        Index := I;
      end { if if with };
end { T_fr_S };

function S_to_T { (S : Str_5;  var Tr : TimeRec) : boolean };
var I : integer;
begin
  I := 1 + scan (length (S), <> ' ', S [1]);
  T_fr_S (Tr, S, I);
  S_to_T := (Tr <> NullTad .T) and (I > length (S));
end { S_to_T };

procedure Yn_to_S { (Yn : YesNo;  var S : Str_5) };
begin
  case Yn of
    Neither : S := '';
    Yes     : S := 'Yes';
    No      : S := 'No';
    Only    : S := 'Only';
  end { cases };
end { Yn_into_S };

procedure Yn_into_S { (Yn : YesNo;  var S : Str_255;
                       Index, Len : integer) };
var Ss : Str_5;
begin
  Yn_to_S (Yn, Ss);
  fillchar (S [Index], ' ', Len - length (Ss));
  moveleft (Ss [1], S [Index + Len - length (Ss)], length (Ss));
end { Yn_into_S };
```

```
  procedure Yn_Fr_S { (var Yn : YesNo;  var S : Str_255;
                      var Index : integer) };
  var Ch : char;
  begin
    Ch := Cap (S [Index]);
    if Ch = 'Y' then  Yn := Yes
    else if Ch = 'N' then  Yn := No
    else if Ch = 'O' then  Yn := Only
    else Yn := Neither;
    if (Yn <> Neither) or (Ch = ' ') then
      Index := Index + 1 + scan (length (S) - Index, = ' ', S [Index + 1]);
  end { Yn_Fr_S };

  function S_to_Yn { (S : Str_5;  var Yn : YesNo;
                      Allowed : YnSet) : boolean };
  var I : integer;  Ok : boolean;
  begin
    Ok := true;
    Crunch_Str (S, S);
    if length (S) = 0 then  Yn := Neither
    else
      begin
        I := 1;
        Yn_fr_S (Yn, S, I);
        if I = 1 then  Ok := false;
      end { else };
    S_to_Yn := Ok and (Yn in Allowed);
  end { S_to_Yn };

  procedure AppendText { (var S : Str_255;  Caps : boolean) };
  var L : integer;  Tmp : Str_255;
  begin
    Crunch_Str (S, S);
    L := length (S);
    if L > 0 then
      begin
        if Caps then  CapStr (S);
        if S [L] = '.' then  S [0] := pred (S [0])
        else if S [L] <> ':' then
          begin
            Tmp := S;
            if not Caps then  CapStr (Tmp);
            if (L <= 18) and
               ((L < 5) or (pos ('.TEXT', Tmp) <> L - 4)) then
              begin
                Tmp := '.Text';
                if Caps then  CapStr (Tmp);
                S := concat (S, Tmp);
              end { if };
          end { else if };
      end { if };
  end { AppendText };

begin { StrOps_U }
  Vv_StrOps_U := Vc_StrOps_U;
  Ck_Version (Vv_Glbs_U, Vc_Glbs_U, Vs_StrOps_U, Vs_Glbs_U);
  *** ;
end {$Q- StrOps_U }.
```

New Library Disk Directory

What follows is a listing of the directory of the newly released Apple disk. The disk was developed by the Apple SIG. See the **Apple SIG Doings** column in the **August 88 NewsLetter** for more information on the naming conventions. Note that the "M" in **APPM???:** stands for "Mac".

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
```

APPMI01:

```
UNITS.DOC.TEXT........ Info on UNITS, SEGMENTS and EXTERNAL Routines
REQUESTS.TEXT......... Requests for programs and routines needed for library
UNIVERSAL.TEXT........ Suggestions for some universal routines for library
USUS.NEWS.TEXT........ News of the formation and early doings of USUS
UCSD.HIST1.TEXT....... Part 1 of the UCSD p-System History
UCSD.HIST2.TEXT....... Part 2 of the UCSD p-System History (in progress)
LIBR.J14.TEXT......... Suggestion for a Library information program
APPLE.DISK.TEXT....... Information on use, packing and shipping disks
APP.SYSP13.TEXT....... Describes the SYSTEMSTUF UNIT and Apple Pascal v1.3 vars
VENDOR.TEXT........... List of UCSD software vendors and products they sell
FILLIN.TEXT........... Fill-in form for information on Orphan Software
ORPHN.SOFT.TEXT....... Information on USUS "Orphan Software" Project
COMAPPMI01.TEXT....... Comments on the above files
DIRAPPMI01.TEXT....... You're reading it
```

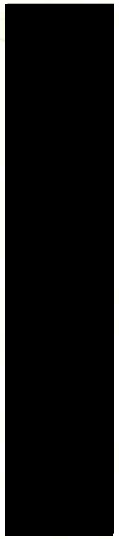
APPMI02:

```
DIR.USAR01.TEXT....... Combined directory list of archive volumes 1-16
SOR.USAR01.TEXT....... Same as above, but sorted by file name
DIR.USAR02.TEXT....... Combined directory list of archive volumes 17-36
SOR.USAR02.TEXT....... Same as above, but sorted by file name
SOR.USARAK.TEXT....... Combined listing sorted by file name   starting A-K
SOR.USARLZ.TEXT....... Same as above                          starting L-Z
APSIG.01.TEXT......... Apple SIG column for NewsLetter of 07/88
APSIG.02M.TEXT........ Apple SIG bulletin sent to members 07/88
APSIG.03.TEXT......... Apple SIG column for NewsLetter of 08/88
APSIG.MAC1.TEXT....... Apple SIG bulletin sent to Mac members 08/88
APSIG.05.TEXT......... Apple SIG column for NewsLetter of 09/88-10/88
APSIG.04.TEXT......... Apple SIG bulletin sent to renewing members
APSIG.06.TEXT......... Apple SIG column for NewsLetter of 11/88
APSIG.07.TEXT......... Apple SIG column for NewsLetter of 12/88
APSIG.08.TEXT......... Apple SIG bulletin sent to renewing members
APSIG.09.TEXT......... Apple SIG column for NewsLetter of 01/89
APSIG.10.TEXT......... Apple SIG column for NewsLetter of 02/89
COMAPPMI02.TEXT....... Comments on the above files
DIRAPPMI02.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
P.O BOX 1148
LA JOLLA, CA 92038

ADDRESS CORRECTION REQUESTED



April 1989

USUS NewsLetter

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

Volume 3
Number 4

NewsLetter Publication Dates

NewsLetter	Due date Code/Forms	Due date Articles	Due date Short stuff
July/Aug 89	06/15/89	06/30/89	07/07/89
Sept/Oct 89	08/15/89	08/25/89	09/01/89
November 89	10/01/89	10/13/89	10/20/89
December 89	11/01/89	11/10/89	11/17/89

Next NewsLetter coming May/June