February 1989

USUS NewsLetter

All the News that Fits, We Print

William D. Smith, Editor

Volume 3
Number 2

From the Editor

by William D. Smith

First, my apologies for being late this NewsLetter. I was too busy to get it done on time.

My thanks to **Bob Spitzer** for contributing a program to clean up text files. *(See page 7.)*

Over the years I have been programing, I have developed a set of units on which I build applications. I call this the WDS environment. I will be describing this environment in the NewsLetters through out the year (as I have time to write). I'm starting off with some background information and a generic quick sort unit in this NewsLetter. I am writing at a medium to good programmer level. If you have questions or comments, please write. *(See page 8.)*

The Chaircritter's Soapbox

by Sam'l Bassett

Now that my name is safely off the masthead of the NewsLetter, I feel up to answering **Adam Rodman's** letter which appeared in the January 89 NewsLetter.

If you will notice the inscription on the front, just underneath the Copyright notice: "**All the News that Fits, We Print**" -- you'll understand why there is so much Board administrivia and so little technical meat in the NewsLetter. Up to very recently, that's all there was, because that's all anyone was writing.

To put it another way, we are just now -- after a year of regular NewsLetter publishing -- getting enough technical stuff in that it's not getting overwhelmed by the administrivia. While I (and indeed the whole Board) heartily agree that more meat and less bureaucratrivia is in order, and is what the membership want to see, we can't write it all ourselves -- we depend on you.

Programmers (and writers) of USUS, write! You have nothing to lose but your anonymity!!

I further agree that there are only a vanishingly small number of members who are interested in the nitty-gritty of Board doings -- this is why we still don't have a Corporate Secretary (no volunteers, and no draftees, either -- they ran too fast), and why, after moving Board meetings to an open section of MUSUS, we have yet to get a non-Board member or non-Officer to show up. There has been an annoyingly large amount of rhetoric about "opening up" USUS and making the decision-making process visible to the membership -- my feeling is and always has been that the membership is basically not interested -- they want technical meat, not parliamentary pyrotechnics.

Board meetings are now held in Room 1 of the MUSUS COnference facility, generally on the second Tuesday of every month. Exact times are announced about one week in advance on MUSUS, and can be found out by inquiring -- I read all the MUSUS sections, and answer questions on-line. Y'All come -- if'n you're interested, that is.

Administrator Says

by Hays Busch

As of January 15th., USUS has 286 active members. Of these 30 are International Members and 9 are Canadian members. Late renewals are still coming in and I feel we will get over the 300 mark by the time you read this. But as you can see, USUS needs to concentrate on getting additional members and holding those we have. This is the number one priority for 1989.

The overall renewal rate was about 35 to 40 percent. Apple SIG members renewed in the greatest number, about 75 percent, to become the largest machine group in USUS, edging out SAGE and STRIDE. Here are the counts by machine.

APPLE (all forms)	81 members
STRIDE	41 members
SAGE	35 members

IBM PCs	34 members
TI 99/4a	27 members
Others	68 members

All of this gives rise to some thoughts I want to share with you. First it is obvious that an active SIG will help hold members in USUS. Frank Lawyer has been doing a great job stimulating interest and activity. Second, the MicroSage Chapter 11 filing may have contributed to the loss of Sage and Stride members. It probably should NOT have since USUS, thru the Sage / Stride / Pinnacle SIG chaired by **Ray Weglein**, is going to be able to support members who still use these machines.

USUS still needs a member to form and take charge of a SIG for IBM PCs and their "work-alikes". And this could become a big SIG since we have a lot of MS DOS based machines in the membership. These are included in the "Others" category above.

A final thought on the figures. It is remarkable how many different machines our members use. I don't have an exact figure here but there are 20 to 30 additional brands represented in the "Others" category. If you own a rather "rare" machine and want to see if any other member has one, let me know and I can advise you. Or hang on for another month when we will publish a complete USUS Roster. We are planning to include a code indicating what machine each member has. Should be quite interesting.

I got to thinking about "mistakes" the other day. (Of course I never make any!!) But now and then I see mistakes happen that I (or USUS) will never know about unless you tell us. For example, one new member did not get a new member packet. Most of the contents of the envelope were returned to the La Jolla Post Office box, but I have no idea who they were sent to. I also mailed a fellow who asked about membership a nice letter and enclosed a membership form. But I forgot to put in the form!

The moral. If you find a stupid mistake in something I or others in USUS do or send to you, don't just mutter those nice compliments under your breath or possibly out loud. Let us know so we can fix the mistake. We try but we are only human and at times even I slip up!

One other thing. If you order SW Library disks or PowerTools, we expect the disk will be readable on your machine and free of garbage. If it is not, let us know. Note I did not say that all the programs on your SW Library disk will run on your machine without some work. There are a few I've never been able to run! Perhaps someday, with help from SIG volunteers, we'll be able to eliminate most of these "clinkers" from the SW Library.

That's all for now.

Treasurer's Report (Dec 1988)
by Robert E. Clark, Treasurer

Bank Balance	$6,443.25	11-30-88
Income - December 1988		
Dues:		(new/renew)
Student	25.00	1/0
General	1,005.00	2/25
Professional	200.00	0/2
Institutional	0.00	0/0
Other Income:		
Library fees	66.00	
PowerTools	206.80	
CIS	0.00	
Misc.	0.00	
Total Income:	1,502.80	
Expenses - December 1988		
Administrator:		
CIS	41.65	
Telephone	40.56	
Photocopies	10.00	
Postage	228.96	
Printing	303.44	
Supplies	67.68	
Other:		
Mail from La Jolla	9.65	
TI-99/4a SIG	59.66	
Bank charge	1.00	
Total Expenses	$762.60	
Bank Balance	$7,183.45	12-31-88

Board of Directors Minutes (Jan. 17,1989)

Due too some unexpected difficulties, the minutes of the January meeting will not be in the News-Letter until next month.

Apple SIG Doings

by Frank Lawyer

PRAISE...

Thanks very much to **Richard S. Murdock** for a keeper copy of the SofTech UCSD p-System Applications Catalog of December, 1982. Incidentally, I am still looking for the date of the most recent of these, if anyone knows... Many huzzahs to all of you who have renewed their USUS membership. (see below).

TRENTON FAIR...

Every year the Amateur Computer Group of NJ sponsors a Computer Fair at Trenton State College just outside of Trenton NJ. This is the granddaddy of the fairs and swapmeets and has been going on for about 12-13 years. They have a huge parking lot full of tables, and several buildings with meeting rooms, classes, demos etc. The fair is always held the 3rd weekend in April. This year a few USUS people have decided to go to the Fair, get a table, sell white elephants, get new USUS members and hang out for the weekend. We have just begun to get organized for this, so no hard details available yet. If you are at all interested, please call me (number below). More details later.

WELCOME DISKS...

The latest version is 02/01/89. We have updated the Contact list to reflect changes as a result of the recent Board of Directors election. Some files were added and the book list and bibliography files were updated. Changes were also made as a result of the APDA move, and the addition of new volumes to the Apple specific library disks. Remember you can always swap for the latest version by sending back your old disks.

LIBRARY DISKS...

I should have mentioned last time that there are two files on disk APP2I03 (listing printed in January issue) which contain the sorted listing of all the directories of the USUS library. These can be loaded into the standard editor and scanned for filenames, dates and whatever. There are two parts, A-K and L-Z, but those of you with ASE (or using the Apple 128K editor) will probably want to combine them into one large file. We'll issue periodic updates as things change. The idea is to make it easier for USUS members to locate library programs. Since they are directories of the "archive" disks, these files should be useful whether you have an Apple or a Sage. The one place I know where there will be a problem is where there were 8080 or Z-80 assembly modules which were not copied to the Apple "archive" disks which were used to produce these lists. In another month or two, we will have 10 or 12 Apple specific volumes available, and I will index their contents the same way.

In working through the "archive" USUS disk library and converting the programs to Apple specific format, one giant need was immediately apparent to me. There was no good way of finding a program which might be useful. Yes, I could look at the names, and yes the welcome disks contain a one line description for (nearly) every program in the library, but if I wanted statistics routines, I might have to hunt a while. A couple of years ago Harry Baya proposed a design for a program which would do the job. Nobody ever implemented it.

Over the last few months I have been thinking about Harry's idea, and I have written a simple exploratory program to provide limited program descriptions, filenames, and a set of keywords which can be scanned in a similar way to the way the BROWSE command can be used on CompuServe. So far, it seems to be working OK, and I'm in the process of gathering the information for the first five or six disks. After I get that information into a file, I'll see how it works and possibly make adjustments. I would like to make this all fit on one, no more than two disk volumes.

The Adventure game is now available on volumes APP2G03 and APP2G04 as of 01/01/89. There are enough files that it takes both volumes. The listings for the directories appear elsewhere in this issue *(page 19)*. This is the original Adventure, with some slight differences from the version that was available some years ago. It plays well, and I have not found any bugs, although I haven't played completely through it. It has a SAVE feature that will allow you to save the game just when you about to be breakfast for a monster.

Also ready as of 01/01/89 is volume APP2U03. This has an IOUNIT which can speed up the use of text files, and SPELLER, a spelling checker. This works very well so far, but I'm still checking it. More next time. If someone knows of an extensive dictionary in TEXT file format, please let me know, because I would like to add words to the limited dictionary that SPELLER uses.

I still need volunteers to help with the conversion process. Don't be shy.

WE'RE NUMBER ONE...

It seems the good rate of renewals by all you Apple SIGgers has put us over the top. I'm told that we have now passed the Sage/Stride bunch and are narrowly in the lead in terms of sheer numbers. Congratulations to us! Now just because we're not second anymore doesn't mean we don't have to keep trying harder.

APDA...

Things weren't quite as bad as I was originally led to believe. Apple Computer will be distributing the Apple software development packages themselves, still using the APDA name. All current APDA members received a new application which will make them members in the new organization. It appears that the fees stayed the same, i.e. $20 per year. Apple is continuing membership until your normal expiration date, so no money is required. All that current users had to do was to sign a new customer agreement to the terms of use.

Apple has established a new number for information at 1-800-282-APDA. Or write them at APDA, Apple Computer, 20525 Mariani Avenue MS 33G, Cupertino CA 95014. Sue Espinosa is Director of Developer Channels. Also Apple set up an APDA booth at MacWorld, so maybe we will have a report from William Smith *(I didn't stop by the Apple booth).*

Whether or not APDA will continue to distribute all the products that were distributed by the A.P.P.L.E. Co-op is uncertain. Apple made no statement of policy as to this. We will know this, and the pricing of the products, when we receive the first APDAlog. I checked the availability and price on Apple Pascal v1.3 and it is still available, at an increased price of $100, up from $75. I was told that so far no products have been dropped, but some products have had price advances. None have decreased. I am also concerned about whether or not these products will have any support, as many were not being supported by Apple prior to this change.

As for the A.P.P.L.E. Co-op, they will still have an organization devoted to distribution of third party software. This will be called TechAlliance. They are giving all current APDA members a complimentary membership. You can contact them at at 290 SW 43rd Street, Renton WA 98055 (206)-251-9798.

BOOKS...

Tom Swan has produced a whole range of books for the UCSD Pascal enthusiasts. While Tom has developed the code and written the books largely around the Apple system, this code should be quite portable to other hardware. Here are three of his efforts.

Pascal Programs for Database Management. Tom Swan. Published by Hayden Publishing Company, Hasbrook Heights, NJ. 1984. 288 pages. $21.95 (paperback) ISBN 0-8104-6272-9.

Tom Swan is the author of some excellent Data Base Software. The book includes programs and library units which make up a UCSD Pascal relational data base called the Pascal Data Base System (PDBS). Complete source is included in the book and is also available on diskette (Pecan). It is Apple specific, but differences for IBM version (Softech version IV.1) are explained in detail. This system has also been ported to other hardware. There are some stand-alone routines you can type in from the book, but it will be better to get the disks and save the time.

The following comments are from **Hays Busch**:

> "This is not a textbook. Contains all source code required for a very good relational data base. Excellent programming style and splendid example of self documenting code and lucid documentation. Many useful basic programming tools which can be adapted to other uses. Any serious programmer should study this book to learn how to write straightforward but elegant code. Requires an intermediate level of UCSD Pascal knowledge."

This book is available from Pecan software, complete with the disks.

Pascal Programs for Games and Graphics. Tom Swan. Published by Hayden Publishing Company, Hasbrook Heights, NJ. 1983. 214 pages. $15.95 (paperback) ISBN 0-8104-6271-0.

Another of his excellent books. This one contains all the source code you need for some good graphics routines and stand-alone programs. The XTRASTUFF unit can be installed in your SYSTEM.LIBRARY to provide a number of graphic functions. Unfortunately, there does not seem to be a source code disk available for this book. Programs are Apple specific. Requires intermediate knowledge to get the most out of it, but Tom's style and thorough documentation mean you can do well with these routines even if you are a relative novice.

Pascal Programs for Business. Tom Swan. Published by Hayden Publishing Company, Hasbrook Heights, NJ. 1983. 224 pages. $21.95 (paperback) ISBN 0-8104-6270-2.

The third of a trilogy. The business programs include some statistical analysis programs, a spreadsheet, a text-formatting program, sorts, charts and others. Programs are Apple specific, but should easily port to other UCSD based environments. Again, I do not think a disk is available.

If anyone knows that disks are available for the latter two books, please tell me where they can be obtained. Thanks.

SUN...

The new Sun Remarketing catalog is out. This catalog is for 1989, and contains all their products. In some cases of software products, the description is not as long as in previous catalogs. You probably already know that Sun supports orphan Apple machines such as the Apple2+, Apple3 and Lisa. They have a variety of hardware and software products which will be of interest. Call 801-752-7631 or write them at PO Box 4059, Logan UT 84321 for the new catalog. A much more complete write-up is on the Apple welcome disk. I have found that they are pleasant to deal with, and their shipping is efficient.

CORRECTION...

Last time I mentioned Pre-Owned Electronics and said they had parts for your Apple, and some used machines too. Since then I have been in contact with them and they are no longer carrying used Apple3s, although they do still have 2+, 2e, and various Macs. They also have a smattering of used MS-DOS type machines if you are in the market.

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

Mac SIG Doings

by Frank Lawyer

This starts a new column for and about Apple Macintosh, the p-System on Mac, other Pascals and Modulas, and generally whatever interests you Mac fans out there. The information in this column is intended to be Mac specific, so I will not repeat here anything which is contained in the **Apple SIG Doings** column.

USUS Mac members have been an under-privileged bunch. USUS has no Mac library volumes, no Mac welcome disk, and little Mac news. Hopefully, we will now change all that.

First, I'm sorry it has taken so long to get moving. This is strictly a function of available time and my desire to have the Apple SIG be on a solid footing. While there is still plenty to be accomplished in Apple2 land, I feel we should now start the Mac SIG on the same road. I learned a lot in working on the Apple SIG, so that experience won't go to waste. Of course, I can use help!

If you own a Mac, you're a Mac SIG member. That eliminates the red tape. However, if you did NOT indicate Mac as your primary machine when you renewed your membership, I may not be aware that you have one. So you will have to send me a note or give me a call. I'll maintain a list. That way, when I send out a Mac bulletin, you'll all get one.

Back in August 1988, I sent out a bulletin to all USUS Mac owners I knew of, even though you might have signed up as IBM owners. I asked (begged) for feedback and ideas, but only received a few replies. While welcome, they were not as informative as I had hoped.

Therefore, in getting started with the SIG, I may have chosen certain paths which may seem downright worthless to you. That's to be expected. Give me your opinions and we'll see what we can do. We have to start somewhere.

One thing which emerged from talking to USUS members who have Macs, is that many of you are not actively using the p-System on them. There are probably a variety of reasons for that. Good compilers and working environments are available from other sources, including Apple and APDA. Apple's chosen development environment for the Mac is NOT a variety of the UCSD p-System.

It seems that we should expand our horizons and encompass other language implementations. UCSD source code is portable to other Pascals with varying degrees of effort. We will have to discover what the most popular implementations are, and try to support them. How can we write portable code that can easily be moved from one language implementation to another? You will have to help.

In the paragraphs to follow, I will outline some of the various choices I have made, and why I made them. Nothing is cast in concrete, so if you have opinions on these, please speak up. As usual, my address and phone number are below.

DISK FORMAT...

The "universal" format still appears to be 400K MFS type disks. I am frankly amazed at this, since the older Macs which used this format didn't have enough power to run most of the commercial software today. If I were a commercial software distributor, I guess I would use this format just so everybody would be able to read it. However, it annoys me to waste half the space on a 3.5 inch disk, so my chosen format for all Mac SIG disks will be 800K HFS. I feel that by this time, many older machines have been converted to Mac Pluses and will be using the "new" format disks. If you MUST have the old format, please so indicate on your correspondence or disk order forms!

WELCOME DISKS...

We have a Mac welcome disk! As a first effort, I converted all the files from the Apple2 welcome disks. These files contain information for all Apple machines, and that made it easiest for me. As we progress, more Mac specific information will be added and the old stuff will be modified. Your ideas will help. The purpose of the welcome disk is to introduce USUS members to the organization, provide a list of resources for your hardware, and facilitate the use of the USUS program library. The library provides a considerable resource of source code that can be used directly on the p-System or can be ported to other Pascal and Modula-2 dialects.

All previous welcome disks of other hardware have been in p-System format. The implicit assumption was that you had and used p-System and could read the disks. Another choice had to be made here, and I have departed from the old format. The USUS Mac welcome disk is in MacWrite format. I felt that most word-processors will at least READ MacWrite format. Another possible choice was plain vanilla ASCII text files, which you could read with almost any word processor as well as a lot of DAs (Desk Accessories). The major reason that I didn't do that was the conversion format. It appears that in the process of conversion, a carriage return was inserted at the end of every line. This was not my intention, and it makes the plain ASCII text files harder to read. I may solve this problem later by another conversion method. Let me know your opinion.

A Mac welcome disk will be in the mail shortly to all of you who renewed and said the Mac was your primary machine. We promised one with renewal and you will get one! For the rest of you with Macs who want the Mac format welcome disk, just slip a 3.5 inch disk in an envelope and send it to me. These plastic shells are a lot more hardy than the 5.25 inch floppy, and they cost less to mail. Wrap it in a plain white piece of paper (or a letter), and you can probably still send it for 25 cents. If you want the 400K format, send two disks and remind me. USUS pays the return postage.

The Mac welcome disk is a renewable resource. We will be making improvements as we go along, so this won't stay static. Remember you can always swap for the latest version by sending back your old disk or a blank one.

LIBRARY DISKS...

The Mac welcome disk contains listings for all of the USUS "archive" volumes. These volumes contain the source code for a large variety of programs. Much of it is very instructive for the beginner or novice programmer and will repay the effort to study them. The good news is that the disks are finally available. Previously, you would have had to convert the volumes to Mac format yourself. The bad news is that not much of this will run on the Mac without at least minor work. If you want to start right in on this, there is an order form file on the welcome disk. Just print it out and order what you want.

At present, there are no Mac specific disks, but there will be shortly. I will begin editing and moving programs from the general USUS volumes to Mac volumes. At one time, so I am told, there was a USUS distributor who distributed the general volumes in Mac format. If anyone has any of those disks, please let me know, because it would ease the burden a little. Or, if you have converted some programs on your own, I could use your experience. I am in the process of doing this same thing for the Apple2 series, so it will be familiar. Nevertheless, I could use some help, so if you would like to contribute some time to USUS, I have a plan laid out.

Transferring these to Mac disks, and compiling and running these on a Mac p-System will only be a start. The programs don't use any of the Mac pull-down menus etc, so there will still be a lot to be done. The Power System permits calls to the toolbox routines, so we can enhance the existing programs to be more Mac like. We will make the code available on p-System format disks, and also ASCII text file disks, so that those of you who prefer Lightspeed Pascal, TML or MPW Pascal, Turbo Pascal etc. can import the text files and have a start. We cannot do everything at once, so bear with us, it is going to take some time. Let me know if you have some ideas on this.

WHAT CAN BE DONE NOW?

I would like to hear your constructive complaints, ideas, and questions. My address and telephone are below. I only ask that you don't call after 9:30PM Eastern time. I am also willing to address specific problems or questions on the Mac versions of the p-System, Pascals and Modula-2s. I sure don't know everything, but I might be able to find out.

What can you do? I thought you would never ask! Write articles, send programs or code sections illustrating some aspect of programming. Don't worry if you are a beginner, everyone was a beginner at some time. The NewsLetter will appreciate your contributions, and so will other USUS members. If you feel you would rather send things to me, that's fine, I can use the material in the column. If you have questions about UCSD p-System, Pascal, Mac etc., I would rather that you send them to me. If there are duplicates, or similar ones, I can combine them.

It just so happens that William Smith, the NewsLetter editor has a Mac II that he uses to prepare the NewsLetter, so you can send him machine readable text *(MicroSoft Word or plain text format)* that he won't have to retype.

I think the best way to go about getting the Apple Mac members involved is "them that does, gets". This means that if you take a chance and start to participate, exchange ideas and put in some effort, you will be rewarded. If you sit back and watch the bulletins go by, that's OK, but it won't be as rewarding, and it won't be much fun either!

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

DeTab (a program to clean up text files)
by Robert Spitzer

Many folks may be familiar with porting text files between environments. For example, the PDOSTRANS program (source on MUSUS) allows DOS files to be imported into p-System. This is very useful when one's secretary uses some godawful word processor under DOS, but you (I) use a Stride for real computing, and by default for word processing. I found in importing text files created by my secretary or students that their WP's loaded the files with all sorts of (unprintable #$%@) junk characters. While the following kludge is not elegant, it brute-force removes the non-printing characters and substitutes spaces, so that ASE won't choke on the files.

For the "student", the program also illustrates the use of "blockread/ blockwrite" to access untyped files regardless of their contents. This is quite useful when dealing with files not created by the p-System text mechanism, or when dealing with files not created as Pascal records. Eventually, it becomes easier and worthwhile to have TOTAL control of the internal structure of a file, a facility pioneered in UCSD Pascal with block-I/O on files. The program also illustrates the representation of characters as bytes, and the capability of manipulating ASCII codes as numbers.

Conversion of this program to strip characters with ASCII values greater than 127 of their non-printing attributes is left as an exercise for the student. (Hint: the answer has been posted as a new file, "DETAB.TXT", in Lib 3 of MUSUS).

WDS Environment (background)
By William D. Smith

Since I'll be including many of my units and programs in future issues, I'll take this time to go over the environment I work in and the way I set up my files. I use the p-System editor, ASE and many other tools I've written to prepare my programs for translation (compilation, assembly, etc.). Not all the programs I write are compiled on or for the p-System. Many of them are uploaded to main-frame computers (Vax, Cyber, Prime, etc.) and compiled and executed there. I find it much faster to do all the editing with ASE rather then trying to learn a different editor for each system. Also I have all the text preparation tools I need in a form that I like best (since I wrote them myself).

The first part of each file contains a header with some file related information, a description of the program, unit, document, etc. and a change log. Since I use ASE for my p-System editing, some of the header information is set up to use its features. Some of the information is used by my tools, and the rest is documentation for the user. Look at the first part of the quick sort file as you read the next few paragraphs.

The first line consists of several parts: the first two characters -- the open comment symbol for the programing language used in the file, a very short comment, the version of the file in square brackets, three dashes, the last changed date, and the ASE macro to change that date.

The first two characters are use to open a comment so that the compiler will work. They are also use to determine the "language" of file. I use the following conventions:

'{ '	Pascal file
'(*'	Modula II file
'--'	Ada file
'/*'	C file
'; '	Assembly language
'. '	document file
'C '	FORTRAN file

An example of a tool that takes advantage of this is a program to strip comments from files prior to uploading for compilation and testing. (There is no use uploading comments when the file is never viewed or printed on the main-frame while it is being developed. This saves in communication time and cost.) The program looks at the first two characters to determine how to locate and handle comments.

ASE has the ability to display a directory when you edit a file or copy from a file. After displaying the directory, you can press the '?' to get the first line of each file displayed. On an 80 column terminal, everything up to the first closing comment is displayed. This lets you quickly compare files, check the version or the date the file was last changed.

The three dashes, '---', are use to indicated whether the file is in the process of being changed. When more then one person can modify a file (or when you are working by yourself and may not get it finished in a short time), the rule is: put your initials where the dashes are, copy the file, make changes to the copy, test, replace your initials with dashes, and copy the file back discarding the original file (after backup). If the file contains initials instead of dashes, then someone is changing the file (or you never finished some changes) and you must treat it as read only. I have used this with both team and single person projects and it works well. Note that all the team members must follow the rule. I have started to modify a file, been side tracked (the modifications were not that

continued on page 10

```
program DeTab;

{ Originally for removing TABs, expanded to other nonprinting chars }
{$C Copyright (C) 1988 A. Robert Spitzer, MD }
{ Permission granted for personal use by all USUS members }
{ Not for commercial sale except as included in packages sold by USUS }

var   I, J          : integer;
      InFname       : string;
      OutFname      : string;
      InFile        : file;
      OutFile       : file;
      InBlock       : packed array [0..511] of 0..255;
      OutBlock      : packed array [0..511] of 0..255;
begin
  writeln ('Non-printing character removal utility');
  writeln ('A. Robert Spitzer MD.  v1.1   10 Mar 88');
  writeln;
  write ('Enter file to clean up: ');
  readln (InFname);

  writeln;
  write ('Enter file name to create: ');
  readln (OutFname);

  reset (InFile, InFname);
  rewrite (OutFile, OutFname);

  J:= 1;

  while blockread (InFile, InBlock, 1) = 1 do begin
    for I := 0 to 511 do begin
      if (InBlock [I] < 32 { control characters }) and
         (InBlock [I] <> 13 { return }) and
         (InBlock [I] <> 0) then
        begin
          OutBlock [I] := 32 { space };
          write ('*')
        end { if }
      else
        begin
          OutBlock [I] := InBlock [I];
          write ('.')
        end { else }

      if I mod 64 = 0 then  writeln;
    end { for };

    writeln;
    write (J);

    J:= J + 1;

    writeln;

    if not (blockwrite (OutFile, OutBlock, 1) = 1) then
      write ('Error writing block!..')
  end { while };

  close (OutFile,lock);

  writeln;
  writeln ('Conversion complete.');
end { DeTab }.
```

WDS Environment

(continued from page 8)

important) and went back a year later to make different changes, and saw that the file was still in a flux state. This told me I would either have to finish the previous modifications or discard them.

The last part of the line is the date the file was last changed and the macro which changes that date when the file is opened. (See page 10, July 1988 NewsLetter for a description of the macro.)

The second line is for the p-System only and just turns off writing messages to the screen while the file is being compiled. The third line consists of a copyright notice which will be embedded in the executable version of the file. Not all compilers support this and the command ($C) may change for each compiler.

The next sections give information about the file, the author, restrictions on the file, the system(s) the program is designed to run under, the compiler(s) which will compile the file, some keywords which describe the file, a description of what is in the file and a change log. Although I have not written it yet, I have plans for a tool (program) which will extract all this information and store it in a database.

The version and date on the line with the file name is that of the last working version of the file. If the file is in a state of flux, this version number is not the same as that on the top line (the version on the top line is the version of the current file, finished or not).

In the author section I always include a way of being contacted and require any programmers who work for me to do the same. This tends to force them to take some pride in their work and thereby produce better programs. More then one author can be listed and anyone who changes the file should have their name included.

The change log is maintained by ASE (set a marker, '$LOG' at the beginning of the first log entry) in reverse order (most recent first). Each entry includes the date (supplied by ASE), the initials of the person doing the change, the version and a comment.

Generic QuickSort Unit

By William D. Smith

The unit on the following pages and the test program following that, demonstrate a way of writing a generic unit to do sorting. They also show several other useful programing hints.

When you write a unit, always write a test program to go along with it (actually its good practice to write the test program first).

The line immediately before the unit name (for the quick sort unit) or the program name (for the test program) is a compiler command to include the file named. On all the p-Systems which I compile programs, I have a 'VERSION.TEXT' file which defines that system. Some of the things in the version file are the definitions of some conditional compilation flags which define the system (ie. AOS, IV13, IV21 and IV22). It also includes a line which tells the compiler where I keep the 'UTIL.LIB'. All my units are contained in this one library. This way I can move the file to a different system and compile it without making any changes to the file being compiled.

When I write a unit, I always put a comment after the interface declaration as follows:

interface {$ Q_Sort_U [1.00] 30 Mar 88 }.

This comment is embedded in the interface section of the compiled unit. When I use Adv-Patch to examine the file, I can easily tell where the interface section starts and what the unit name is. The '$' is needed so that my tools don't strip or modify this comment.

My units also include the two constants 'Vc_<unit name>' and 'Vs_<unit name>' and the variable 'Vv_<unit name>'. These three values are used for version control at runtime. Read 'Vc_' as version constant, 'Vs_' as version string, and 'Vv_' as version variable.

In the declaration of the procedure 'Q_Sort' you see the word **interface**. This tells the compiler that the variable's size will be determined at runtime. The two parameters in the brackets ('Ml' and 'Mh') are required and must be value parameters declared as integers. They are not specified in the call (see the main body of 'Test_Q_Sort_U'). The compiler generates code

to determine these parameters and put them on the stack. You may use them in the called procedure. The first is the lowest index of the array and the second is the highest index of the array as declared in the procedure. The array 'A' to be sorted is passed to the procedure as an array of integers. The procedure figures out how large each element is from the 'Bytes' parameter. This is needed for determining the actual indexing of the array. The 'Save' parameter is a one element working space used in the 'Split' procedure.

Now the other important fact is how you write the greater then or equal function ('Geq'). The function must be declared as follows (the name of the function and parameters may be different):

```
function GeqAi (var Src, Tgt :
          integer) : boolean;
```

Since 'Src' and 'Tgt' are declared as variable parameters, the compiler passes their address on the stack when they are called. That is, 'Src' is the address of the first word of the source element in the array. If the array being sorted consists of one word elements like integers, you can do a simple comparison as in 'GeqAi'. If the elements are larger then one word, you have to copy the address passed on the stack into a pointer to the type of element you need using the pmachine instruction as follows:

```
pmachine (^Ts, (Src), 196 {Sto});
{ Ts := Src; }.
```

'^Ts' pushes the address of 'Ts' onto the stack. '(Src)' pushes the value of 'Src' (which is itself an address) onto the stack. The p-code STO (value 196 in version IV) stores the value at the top of stack ('Src') into the address at top of stack minus 1 (address of 'Ts'). Now 'Ts' points to a record in memory. The array sorted in this case is an array of pointers (see 'Ap' and 'GeqAp'). If you were sorting an array of records (see 'Ar' and 'GeqAr') or packed records (see 'Cr' and 'GeqCr'), '(Src)' would be changed to '^Src'. This would put the address of the element in the array on the stack. When the pmachine instruction is complete, 'Ts' points into the array. When you use the pmachine instruction, **always** include the comment which shows the pseudo code.

```
{ WDS Quick sort unit                [1.00] --- 30 Mar 88 } { |xjm$d|nx|f8|e|. }
{$Q+}
{$C (c)  William D. Smith  -- 1988, 1989,  All rights reserved.                }

{ File:          Q_Sort_U.Text            Version 1.00    30 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) 1988 to 1989.

  System:        Power System version IV.2.2

  Compiler:      Power System Pascal Compiler

  Keywords:      WDS Q_Sort_U Quick Sort Unit

  Description:   Quick sort unit.  This unit contains a non-recursive quick sort
                  procedure which can be used by any program.  It will sort any
                  non-packed array (I hope) if the "greater then or equal" (Geq)
                  function is correct.  "Geq" gets var parameters of type integer
                  which are really addresses to the first word of the array
                  element.  The client must set up the "Geq" function correctly.

  Change log: (most recent first)
```

```
Date       Id   Vers  Comment
---------  ---  ----  -------
30 Mar 88  WDS  1.00  Changed from include file to unit.
}
{$I VERSION.TEXT}  { Declares conditional compilation flags }
unit Q_Sort_U;
interface  {$ Q_Sort_U [1.00] 30 Mar 88 }
const    Vc_Q_Sort_U = 1;  { 30 Mar 88 }
         Vs_Q_Sort_U = 'Q_Sort_U';
var      Vv_Q_Sort_U : integer;
  procedure Q_Sort (var A     : interface { the array to be sorted }
                                  array [Ml..Mh : integer] of integer;
                    var Save  : interface { work area, an array element }
                                  array [Lo..Hi : integer] of integer;
                        First : integer;   { first element sorted }
                        Last  : integer;   { last element sorted }
                        Bytes : integer;   { size of array element in bytes }
                    function  Geq (var Src, Tgt : integer) : boolean);
                                          { Src >= Tgt function }
implementation
  procedure Q_Sort;
  var    Words : integer;  { size of array element in words }
         Index : integer;
         Tos   : integer;
         Stack : array [0..10 { 32K elements }, 0..1] of integer;
    procedure Split (First, Last : integer;  var Index : integer);
    label 1;
    begin
      Last := Last + Words;  { move Last by the size of one element }
      moveleft (A [First], Save [Lo], Bytes);  { Save := A [First]; }
      while Last > First do begin
        repeat
          Last := Last - Words;
          if First = Last then  goto 1  { exit while loop };
        until Geq (Save [Lo], A [Last]);
        moveleft (A [Last], A [First], Bytes);  { A [First] := A [Last]; }
        repeat
          First := First + Words;
          if First = Last then  goto 1  { exit while loop };
        until Geq (A [First], Save [Lo]);
        moveleft (A [First], A [Last], Bytes);  { A [Last] := A [First]; }
      end { while };
    1:
```

```
      moveleft (Save [Lo], A [First], Bytes);  { A [First] := Save; }
      Index := First;
    end { Split };

  begin  { Q_Sort }
    Words := Bytes div 2;
    First := M1 + (First * Words);
    Last := M1 + (Last * Words);
    Tos := - 1;  { empty stack }

  { Push (First, Last); }
    Tos := Tos + 1;
    Stack [Tos, 0] := First;
    Stack [Tos, 1] := Last;

    while Tos >= 0 do begin
    { Pop (First, Last); }
      First := Stack [Tos, 0];
      Last  := Stack [Tos, 1];
      Tos := Tos - 1;

      while First < Last do begin
        Split (First, Last, Index);

        if Index - First > Last - Index then
          begin
          { Push (First, Index - Words); }
            Tos := Tos + 1;
            Stack [Tos, 0] := First;
            Stack [Tos, 1] := Index - Words;
            First := Index + Words;
          end { if }
        else
          begin
          { Push (Index + Words, Last); }
            Tos := Tos + 1;
            Stack [Tos, 0] := Index + Words;
            Stack [Tos, 1] := Last;
            Last := Index - Words;
          end { else };
      end { while };
    end { while };
  end { Q Sort };

begin { Q_Sort_U }
  Vv_Q_Sort_U := Vc_Q_Sort_U;

  *** ;
end {$Q- Q_Sort_U }.
```

```
{ Test quick sort unit (Q_Sort_U) [1.00] --- 30 Mar 88 } { |xjm$d|nx|f8|e|. }
{$Q+}
{$C (c)  William D. Smith  -- 1988, 1989,  All rights reserved.              }
{ File:          T.Q_Sort_U.Text          Version 1.00    30 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) 1988 to 1989.
  System:       Power System version IV.2.2
  Compiler:     Power System Pascal Compiler
  Keywords:     WDS Test_Q_Sort_U Test Quick Sort Unit
  Description:  This program tests the Q_Sort unit.
  Change log: (most recent first)
Date        Id   Vers  Comment
---------   ---  ----  -------
30 Mar 88   WDS  1.00  Wrote first version.
}
{$I VERSION.TEXT}  { Declares conditional compilation flags }
program Test_Q_Sort_U;
uses    Glbs_U,     { WDS globals unit }
        StrOps_U,   { WDS string ops unit }
        T_Io_U,     { WDS terminal I/O unit }
        Q_Sort_U;   { WDS quicksort unit }
const   Version     = '[1.00]';  { 30 Mar 88 }
        MinA        = 0;
        MaxA        = 31;
type    A_Ptr       = ^A_Rec;
        A_Rec       = record
                        A : integer;
                        B : integer;
                        C : integer;
                      end { A_Rec };
        C_Rec       = packed record
                        A : char;
                        B : Byte;
                        C : Byte;
                        D : integer;
                      end { C_Rec };
var     I           : integer;
        Si          : integer; { Save area for Ai sort }
        Sr          : A_Rec;    { Save area for Ar sort }
        Sp          : A_Ptr;    { Save area for Ap sort }
        Sc          : C_Rec;    { Save area for Cr sort }
        Ap          : array [MinA..MaxA] of A_Ptr;
        Ar          : array [MinA..MaxA] of A_Rec;
        Ai          : array [MinA..MaxA] of integer;
        Cr          : array [MinA..MaxA] of C_Rec;
        S           : Str_15;
```

```
function GeqAi (var Src, Tgt : integer) : boolean;
{ Sort an array of integers }
begin
  GeqAi := Src >= Tgt;
end { GeqAi };

function GeqAr (var Src, Tgt : integer { A_Rec }) : boolean;
{ Sort an array of records }
var Ts, Tt : ^A_Rec;
begin
  pmachine (^Ts, ^Src, 196 {Sto});  { Ts := Src; }
  pmachine (^Tt, ^Tgt, 196 {Sto});  { Tt := Tgt; }
  GeqAr := Ts^ .C >= Tt^ .C;
end { GeqAr };

function GeqAp (var Src, Tgt : integer { A_Ptr }) : boolean;
{ Sort an array of pointers }
var Ts, Tt : A_Ptr;
begin
  pmachine (^Ts, (Src), 196 {Sto});  { Ts := Src; }
  pmachine (^Tt, (Tgt), 196 {Sto});  { Tt := Tgt; }
  GeqAp := Ts^ .C >= Tt^ .C;
end { GeqAp };

function GeqCr (var Src, Tgt : integer { C_Rec }) : boolean;
{ Sort a packed array of records }
var Ts, Tt : ^C_Rec;
begin
  pmachine (^Ts, ^Src, 196 {Sto});  { Ts := Src; }
  pmachine (^Tt, ^Tgt, 196 {Sto});  { Tt := Tgt; }
  GeqCr := Ts^ .C >= Tt^ .C;
end { GeqCr };

procedure Dump (Xx : integer);
{ This procedure displays the fields of the arrays to be sorted. }
var I : integer;  S : Str_63;
begin
  SetXy (Xx, Py + 2);
  W_Str (' I  Ap^ .C  Ar .C   Ai   Cr .C');
  SetXy (Xx, Y + 1);
  W_Str ('--  ------  ------  ---  -----');
  for I := MinA to MaxA do begin
        {12345678 1 2345678 2 2345678 3}
    S := '                                ';
    I_into_S (I, S, 1, 2);
    I_into_S (Ap [I]^ .C, S, 6, 3);
    I_into_S (Ar [I] .C, S, 14, 3);
    I_into_S (Ai [I], S, 21, 3);
    I_into_S (Cr [I] .C, S, 27, 3);
    SetXy (Xx, Y + 1);
    W_Str (S);
```

```
    end { for };
  end { Dump };
begin { Test_Q_Sort_U }
  S := 'Test_Q_Sort_U';
  Ck_Version (Vv_Glbs_U, Vc_Glbs_U, S, Vs_Glbs_U);
  Ck_Version (Vv_StrOps_U, Vc_StrOps_U, S, Vs_StrOps_U);
  Ck_Version (Vv_T_Io_U, Vc_T_Io_U, S, Vs_T_Io_U);
  Ck_Version (Vv_Q_Sort_U, Vc_Q_Sort_U, S, Vs_Q_Sort_U);
  C_Sc (false);
  S_Msg (false, false, CopyRight);
  I := S_Prompt (concat (S, ':'), Version, false);
{ Initialize the fields to be sorted in the reverse order. }
{ Probably should initialize in random order. }
  for I := MinA to MaxA do begin
    new (Ap [I]);
    Cr [I] .C  := 150 - I;
    Ar [I] .C  := 200 - I;
    Ap [I]^ .C := 250 - I;
    Ai [I]     := 300 - I;
  end { for };
  Dump (3);
  SetXy (0, Y + 2);
  W_Str ('Sorting Ai');
  Q_Sort (Ai, Si, MinA, MaxA - 1, sizeof (Si), GeqAi);
  W_Str (', Ar');
  Q_Sort (Ar, Sr, MinA, MaxA - 1, sizeof (Sr), GeqAr);
  W_Str (', Ap');
  Q_Sort (Ap, Sp, MinA, MaxA - 1, sizeof (Sp), GeqAp);
  W_Str (', Cr');
  Q_Sort (Cr, Sc, MinA, MaxA - 1, sizeof (Sc), GeqCr);
  Dump (43);
  Error ('End, all but last element sorted');
end {$Q- Test_Q_Sort_U }.
```

The following are the declarations (from the WDS units) used in the test quick sort program. The units will be covered in another NewsLetter.

Glbs_U

```
const  Vc_Glbs_U  = 5;    { 14 Feb 88 }
       Vs_Glbs_U  = 'Glbs_U';
       CopyRight  =
         '(c) Copyright 1983 to 1989 by William D. Smith.  All rights reserved';
type   Byte       = 0..255;
       Str_15     = string [15];
       Str_63     = string [63];
       Str_255    = string [255];
var    Vv_Glbs_U  : integer;
```

```
procedure Ck_Version (Vv, Vc : integer;  Vs_User, Vs_Used : Str_15);
{ This procedure compares Vv and Vc.  If they are different, an error message is
  output, the procedure waits for a return to be typed and then the program
  halts.  Vv for each unit should be the first variable.  Vs_User is the user
  program or unit name.  Vs_Used is the used unit name. }
```

StrOps_U

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

var    Vv_StrOps_U      : integer;

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 if I is >  Len, then the field is filled with stars.  If Len is
  Null, the integer is left justified. }
```

T_Io_U

```
const  Vc_T_Io_U = 8;  { 09 Mar 88 }
       Vs_T_Io_U = 'T_Io_U';

var    Vv_T_Io_U : integer;
       Y         : integer;  { r/o, current line on the screen }
       Py        : integer;  { r/w, current prompt line, change as needed }

procedure SetXy (Xx, Yy : integer);
{ Set the cursor position to Xx and Yy.  If Xx and/or Yy are Null, they are set
  to zero.  Put the cursor at the position of Xx and Yy and update the global
  values of X and Y.  No checking is done. }

procedure C_Sc (NoMsg : boolean);
{ Clear the screen.  The cursor is left at the home position (0, 0).  If NoMsg is
  true, the message line is also cleared.  This is used for writing to the
  console as if were a file (ie. using Tx_Io_U). }

procedure W_Str (S : Str_255);
{ Write the string S to the screen. }

function S_Prompt (Prompt  : Str_255;
                   Version : Str_7;
                   Reverse : boolean) : integer;
{ Show prompt.  The prompts are written on line Py (global).  Prompt is the
  string written and Version is a string written to on the far right side of the
  screen.  If Reverse is true, the prompt is shown with a S_Rev attribute.  The
  first and last column on the line is used for the attributes.  The function
  returns the X location of the end of the prompt string (where the cursor would
  normally be plus one space).  If Prompt won't fit on the screen, it is
  truncated. }

procedure S_Msg (Prompt, Attention : boolean;  S : Str_255);
{ Show message.  If Attention, the message line is blinking and the terminal bell
  is rung once.  If Prompt, the cursor is left at the end of the message. }

procedure Error (S : Str_255);
{ Error.  This procedure displays the string S on the message line on the screen
  and waits for the user to type a space. }
```

Q & / | A? ...

***Hays Busch** (CIS: 70260,306) sent the following problem he is having with an AT clone.*

I've had a DOS hosted p-System (IV.2.1, [R4.4G]) for some time but not used it much since most of my work is on a Sage. The other day I gave it a "test run" on an AT clone and ran into some very interesting problems with both F(iler and E(dvance.

First, checking B(ad blocks in F(iler on a 1.2 MB floppy, all blocks past 720 were bad.

Next checking B(ad blocks on an old 720 K drive, the report was 6 bad ones from 9 -> 14, 9 good ones from 15 -> 23 then 6 more bad, etc. This pattern kept up until the disk was completely checked with a report of 30 bad blocks. The F(iler is version 6R4.0d.

Now here's the "capper"! That same disk checks OK for B(ad blocks in the high capacity drive. Oh yes, both disks were properly formatted by the drive they were in.

Also E(dvance would not read files on the 720K drive and kept reporting a "bad block 0", if I recall correctly. Did not try E(dvance on the high capacity drive. E(dvance is version 1R2.2.

I know, I know...they're "old" versions and probably should be updated. But has anyone else had this problem, or does anyone else know if it is fixed in an update? Or is my machine doing something nuts?

Except for these "quirks", everything else seemed to work OK and I was pleased to see the familiar p-System running on the AT.

This is "no big deal", but I do copy some USUS files on the AT for some of our members. I think they are OK, but would like to feel better about what I am doing.

***Paul Chidley** (71106,752) posted the following message on MUSUS:*

GRAPHICS - Is that a topic of interest to anyone out there? My friends and I run homemade systems using the 65816 and the V9938 video processor from Yamaha. (same as in the Geneve by Myarc.) Anyway we have written several video programs including an RLE decoder and we're currently working on a GIF decoder. Our machine dependant stuff is in a library so porting to some other machine should only require an equiv unit. HAS ANYONE OUT THERE WORKED OR IS WORKING ON A GIF DECODER? We're also well on the way to converting the old 6502 PME (p-machine emulator) to a generic 65816 version if that interests anyone. (Eg an Apple2 using a 65802)

*An Answer from **Harry Baya** (72135,1667):*

Paul, Since you put the message in this forum, may I assume that you are working in Pascal or a related language? What is a "decoder". Is this a Pascal program that will display a screen from GIF format?. If so I am sure you could find someone interested in porting it to Turbo Pascal and, probably someone to do the same with UCSD for a few machines. I have assumed that some of this graphics stuff tends to be hardware dependent. How portable do you think your stuff is?

Paul Chidley replies:

Harry, yes I'm writing in UCSD Pascal, the only machine code used is the stuff to interface to the machine dependant hardware so yes the code should be pretty easy to port to other UCSD machines. The RLE decoder (ie. display a RLE picture on the screen) is slow, but it is in UCSD Pascal and it works. Yesterday I finished converting a Turbo Pascal program to UCSD that will display MacDraw/MacPaint files, ie. very hi-res monochrome. There are already lots of GIF decoders in Turbo Pascal, I was looking for any already done in UCSD. CompuServe guards the source and I still waiting for approval to access it.

Your editor has a question (CIS:73007,173).

Has anyone ever figured out how to determine if their code (at runtime) is running on a byte addressed or a word address machine? What I need is a Pascal procedure which returns true if the hardware is word addressed (or the outline of the theory for determining this). Using operating system dependencies will not help since I need it to run under several different operating system.

New Library Disk Directories

What follows is a listing of the directory of the newly released Apple disks. These disks were developed by the Apple SIG. See the **Apple SIG Doings** column in the **August 88 NewsLetter** for more information on the naming conventions.

APP2G03:

ADVS1.TEXT	File needed by ADVINIT.CODE to build ADVMSGS, ADVDATA
ADVS2.TEXT	ditto
ADVS3.TEXT	ditto
ADVS4.TEXT	ditto
ADVS5.TEXT	ditto
ADVS6.TEXT	ditto
ADVS7.TEXT	ditto
ADVS8.TEXT	ditto
ADVS9.TEXT	ditto
ADVS10.TEXT	ditto
ADVS11.TEXT	ditto
ADVINIT.TEXT	Program to build ADVMSGS and ADVDATA files
ADVINIT.CODE	Code file for above
ADV.DOC.TEXT	Contains complete documentation for building Adventure
ADV.TEXT	Main TEXT module for Adventure game. Needs ADVSUBS, ADVVERB
COMAPP2G03.TEXT	Comments on programs above
DIRAPP2G03.TEXT	You're reading it

APP2G04:

ADVSUBS.TEXT	Subfile included by ADV.TEXT for compile
ADVVERB.TEXT	Subfile included by ADV.TEXT
ADV.MISCINFO	Tells Adventure your screen size (now set for 24 x 80)
ADV.CODE	Codefile for the Adventure game
ADVMSGS	Messages file for the Adventure game
ADVDATA	Data file for Adventure
COMAPP2G04.TEXT	Comments on programs above
DIRAPP2G04.TEXT	You're reading it

APP2U03:

IOUNIT.DOC.TEXT	Documentation for IOUNIT and IOTEST
IOUNIT.TEXT	A UNIT to quickly read and write TEXT files and strings.
IOUNIT.CODE	Compiled CODE file used to link programs needing IOUNIT
IOTEST.TEXT	A program to test and benchmark IOUNIT
IOTEST.CODE	Code file for above
DICT.TEXT	A small literal dictionary for SPELLER
SPELL.DOC.TEXT	Documentation for SPELLER and DICT
SPELLER.TEXT	A Pascal spelling checker that uses IOUNIT
SPELLER.CODE	Code file for above
COMAPP2U03.TEXT	Comments on programs above
DIRAPP2U03.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

FIRST CLASS MAIL

February 1989

USUS NewsLetter

Volume 3
William D. Smith, Editor Number 2

NewsLetter Publication Dates

NewsLetter	Due date Code/Forms	Due date Articles	Due date Short stuff
April 89	03/01/89	03/10/89	03/17/89
May/June 89	04/15/89	04/28/89	05/05/89
July/Aug 89	06/15/89	06/30/89	07/07/89
Sept/Oct 89	08/15/89	08/25/89	09/01/89

Next NewsLetter coming March