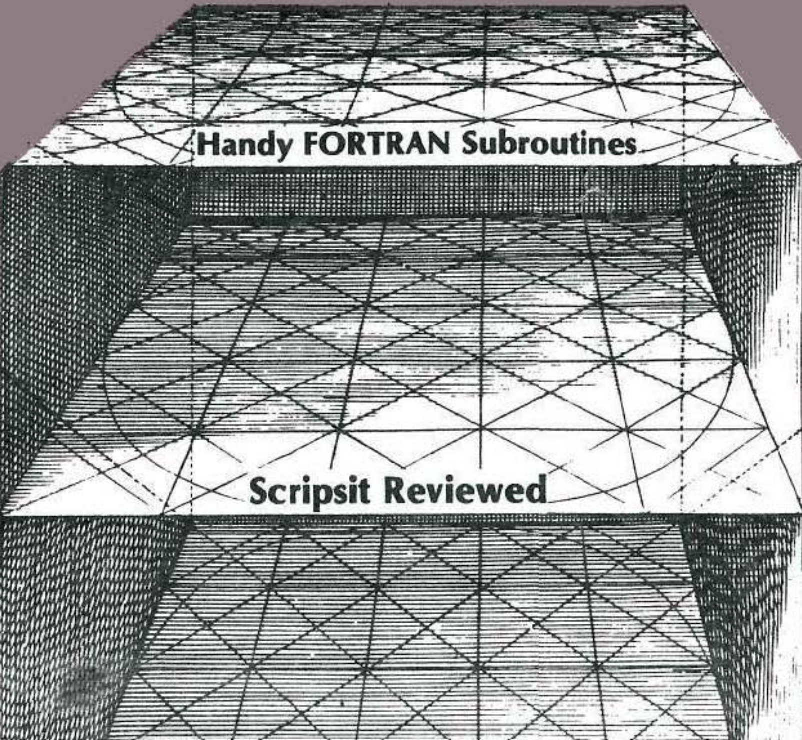
DEDICATED TO THE SERIOUS PROGRAMMER

PROG/80

AUGUST 1980 $3.00



Handy FORTRAN Subroutines

Scripsit Reviewed

Timesharing Column

MORE!

PROG/80

Outstanding Utilities from RACET Computes

INFINITE BASIC

Adds more than 70 commands to BASIC that can be merged in any combination to make efficient use of memory. Includes matrix read, inverse, transpose, identity, simultaneous equations, scalar, vector, and multidimensional array arithmetic, dynamically reshape, expand, and delete arrays, change arrays in mid program, read and write arrays on tape, copy elements, zero and move arrays. String functions include left and right justify, truncate, rotate, text justification, string centering, delete and insert substring, pack string, convert upper and lower case, translate characters, reverse strings, verify function, test number of occurences, masked string searches, encrypt and decrypt string, compress and uncompress string characters. High speed sort routines for strings and arrays, including multikey sorts, are also part of this package.

For business users, an add on package includes multiple precision packed decimal arithmetic, with up to 127 digits of accuracy, binary search of sorted arrays, insert elements in sorted arrays, automatic page headings, footings, and pagination including forced end of page, and automatic hash for record retrieval. **Infinite BASIC $49.95. Infinite business (add on) $29.95**

COMPROC COMMAND PROCESSOR

Chain multiple steps in disk BASIC upon power up, relocatable key debounce, allows pauses for data entry at specified steps during execution (on cassette for disk systems only) for **$19.95**.

DOSORT

BASIC control program with high speed machine language sort for disk users. Merge and sort files on more than one disk if you have 2 or more drives. Self prompting with manual. Specify 32K or 48K version **$34.95.**

GSF GENERALIZED SUBROUTINE FACILITY

18 machine language subroutines with easy access for BASIC users. Sort 1000 element arrays in 9 seconds, read and write arrays to tape, compress and uncompress data, move arrays in memory, duplicate memory, fast horizontal and vertical lines, 5 routines for screen control. Specify 16K, 32K, or 48K version, for **$24.95.**

REMODEL PROLOAD

BASIC program utility allows you to renumber portions of a program, move portions from one location to another, delete, merge, save and verify combined and changed programs, and create your own library of programs, subroutines, and even data statements. Works on tape or disk systems. Two programs on one tape. Specify 16K, 32K, or 48K (unspecified orders receive 16K) tape **$34.95**

PROG/80

VOL. 1 NO. 9

Publisher
Roger W. Robitaille, Sr.
Business Manager
Elizabeth J. Robitaille
Editor-in-Chief
George W. Blank
Editor
Lance Micklus
Software Editor
James Garon
Time Sharing Editor
Richard Taylor
Copy Editor
Sheila Jahnle
Creative Director
Elaine Cheever
Production
Laurie Miller
Gail Nash
Donna Bennett
Subscriptions Manager
Diana Bishop
Customer Service
Bette Keenan
Cartoonist
Sandy Dean

PROG/80 is published bimonthly by SoftSide Publications, Milford, New Hampshire. Telephone (603) 673-5144. Subscription rates: USA regular bulk rate - $15 per year. USA first class, Canada, Mexico, APO/FPO, overseas surface mail - $21 per year. Overseas airmail - $27 per year. **Make all remittances payable in US funds.** Mail Subscription Inquiries to PROG/80 Subscriptions Manager, PO Box 68, Milford, NH 03055.

PLEASE NOTE:

Our WATS line number is:

1-800-258-1790

(In NH call 673-5144)

TOLL FREE
9 AM to 9 PM*
(Monday - Friday)
12 to 4 PM
(Saturdays)

It is ONLY to be used for ordering software.

Programmers will not be available on this line.

Your Cooperation Will Be Most Appreciated

*Eastern Standard Time

OUTGOING MAIL

by Lance Micklus

A number of years ago, I remember hearing about a new product called liquid toilet paper. This was a new type of toilet paper which came on a roll, just like conventional toilet paper, but was soaked in a chemical liquid. The company that was going to market liquid toilet paper felt that the product would be successful because it was superior to the toilet paper we still use even to this day. Its advantages were that it cleaned better, disinfected for better sanitation, contained moisturizers to promote healthy skin, and perfume to control odors.

Around 1965, a man I knew in Hyde Park, New York worked for a new company that made cardboard boxes. From time to time, people with new products to market would come to him to find the best way to package their product. One such product was a can of air. You just pressed the top, and out came a blast of air. When I asked what somebody would do with a can of air, he suggested that I might try it next time I needed to blow the whiskers out of my electric shaver.

In order for a product to exist in the market place, it must meet certain criteria. First, there must be a need for the product. Second, the product must satisfy the need. And third, the product must satisfy the need better than any other alternative.

I don't think I need to justify the importance of toilet paper to our society. Liquid toilet paper satisfied the need for a product to do the things people normally do with toilet paper. So, the first two criteria were met. But it failed on the third test - and for a very interesting reason - there was no way to espouse the virtues of liquid toilet paper on television. Thus, there was no way to convince the consumer that it would satisfy his needs better than other alternatives.

The can of air was a product which posed no advertising problems. It failed as a product because it could not find a real need for its existence.

Maybe you think I'm joking about these two products. I'm not kidding you because I have purchased both products.

If you've ever had to change a baby's diaper, you know what a mess you can sometimes find. Several years ago, moistened towelettes came on the market. Basically, these were really disposable wash cloths. But they really were liquid toilet paper repackaged. The product is so successful that several companies now manufacture it.

The fact that this product sells well is only logical. First, there is a need to clean up the baby's messy bottom. Second, the product satisfies the need. Third, and most important, it does it better than the alternative which is a wash cloth. It's easier to advertise baby products than personal products for adults. This means that the message can be sent to the consumer whereas before it was difficult.

There are also several companies selling cans of air. One brand name which I used is called OMIT. When I was doing a lot of dark room photography work, I used it to blow the dust off of my negatives. It is better than blowing the dust off with your mouth because the blast of air is stronger and there is no danger of accidently spitting on the negative.

Again, the three criteria are met and the product sells well. Photographers need a way to blow a strong but clean blast of air on the negatives and the lenses of their cameras. OMIT does this, and does it better than any other method.

The three criteria also apply to services as well as products. Take pay TV as an example. Several experimental pay TV systems were set up in the late 1960's but they failed. The feeling among experts was that this potentially could become a big market. The feeling of the public was, "Who needs it?". All the movies they wanted to see were on commercial television for free.

In 1972 Time-Life entered the pay TV market with a service called Home Box Office. Today, HBO is the largest pay TV service in the world. Its success, where others failed, isn't hard to understand.

Time-Life's entry into this market made notice. They are a very large company which doesn't enter into a new market unless they really feel there's something out there. When Time-Life moves, people watch.

Time-Life brought to the pay TV industry its big financial muscle. Equally important, it also brought in its massive marketing skills. Together, this put Time-Life into the position of being able to put it all together as a package.

But this does not help satisfy the three criteria; missing is the need for the service upon which the remaining two criteria are then tested. Time-Life did not create the need. Rather, they were in the right place at the right time. The need was created by a change in the way the subjects in motion pictures were treated. This included the showing of nudity, portraying violence very realistically, and scripts which could use any word in the English language. Commercial television was not - and still is not - ready to accept this type of material. The solution was for commerial television to edit these films to meet their standards. The result often is a different movie.

This change brought about the need for a way to show movies on television which were not edited in any way. Capitalizing on this need, HBO built its Home Box Office movie network. Not only did HBO satisfy the need and do it better than the alternatives, it also added icing to the cake with specials such as TIME WAS and THE NEW CANDID CAMERA. The important point to remember here is that these specials do not satisfy the need, but they do help to make the solution more attractive than the other alternatives.

I see many parallels between the pay TV industry and the paid computer access industry like MicroNET and The SOURCE. It strikes me that here again, we have a service with great potential looking for some great need to satisfy. So far, the users of these services are using paid computer access for a variety of indivudual reasons. A lot of users are just hobbyists, doing it for fun. Others just want to be part of the action. Finally, there are a few special groups that have some very specific needs like access to market prices.

Radio Shack's entry into this market is like Time-Life's entry into the pay TV market, and everybody is taking notice because it is Radio Shack. They have the financial muscle and marketing skills to put the package together and to sell it. The question is, "What are they going to sell?".

It appears that Radio Shack is looking, not at the needs of the hobbyist or special interest user, but at John Q. Public as a whole. They have identified certain needs and feel that the paid computer access service they are offering will meet these needs. So far, I agree. The third criteria is that they must meet these needs better than the alternatives.

I'm trying to imagine myself trying to get my mother to go out and buy Shack's video terminal for $400, and using it at $5.00 per hour plus toll charges. Living in Clifton, New Jersey, 10 miles from New York City, she has access to two 24 hour all news radio stations, one 24 hour all news cable television network, frequent news broadcasts on commercial television, New York City's weather radio station, and more newpapers and magazines than you can count. My mother doesn't care about market prices and stuff like that. She is well known in Northern New Jersey for her work with Friends of Animals. If she has any need at all, besides paying the bills, it would be the humane treatment of animals by people. We must concude that paid computer access fails to meet her needs, or if it does, it is a poor alternative compared to other solutions. I don't think my mother is unique in this respect. Maybe your mother has other interests, but imagine yourself trying to talk your mother into buying a video terminal and using it.

I think in the long run, we will see the day when John Q. Public will use some form of paid computer access. The way we will get there may be by a different path than many see now. The logical thing to do is to apply the three criteria and see where it takes us.

The first step is to identify the need. Since the largest single group of paid access users are hobbyists, I would look at their needs and address those first.

The SOURCE is doing a very poor job. It is suffering from hardware problems that were supposed to be cured months ago. This makes it difficult or impossible to use what services it does provide to their fullest potential.

MicroNET, in my book, was an even bigger disappointment. I was under the impression that MicroNET was using a DEC20 system. Having used MicroNET, I'm beginning to wonder. They have done a fine job of ruining TOPS20, the standard monitor on the DEC20 (equivalent to TRSDOS on the TRS-80). Look, everybody, the world does not need another unique operating system. We have enough right now. Besides, TOPS20 is perfect for a lot of things hobbyists want to do. It includes an excellent mail feature, TALK, as well as group owned and system owned file structures. You can even create your own new accounts. If MicroNET is using TOPS20, then they have stripped it of many of the features most of their users need. There are only two things I have found on MicroNET that do address the needs of the hobbiest: MNET-80 (the TRS-80 users group) and CB.

As far as I'm concerned, the current needs of the paid access user are not being satisfied as well as they could be. In addition, Radio Shack's control over MicroNET has now created a new need - a paid computer access system NOT controlled by Radio Shack. Certainly, your neighborhood Apple and PET dealer is not going to want to send his customer down to Radio Shack. Nor will users want to continue to use MicroNET when they find out that Radio Shack can access their files and distribute the contents of those files throughout the company.

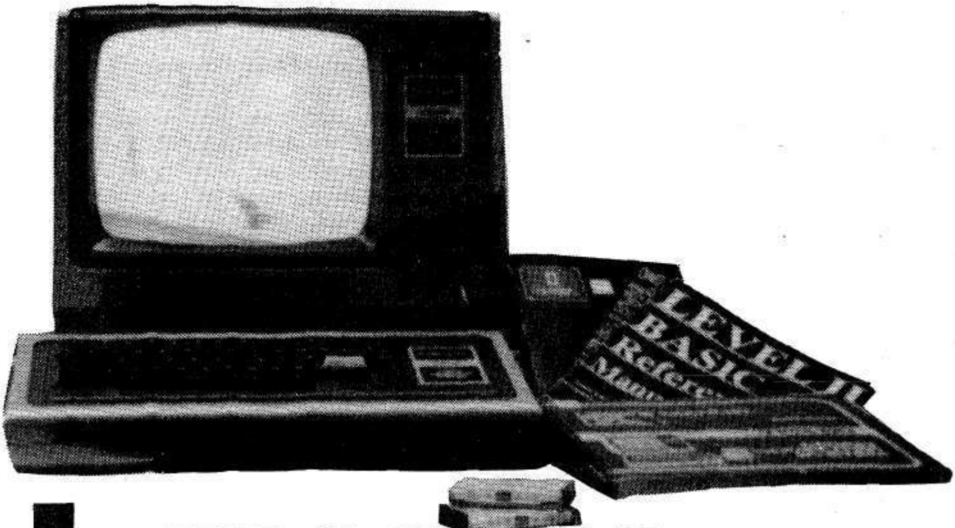
I think the needs are best satisfied by a third and new paid access system using a couple of DEC2060's tied together. Things like CB, bulleting boards, and data bases are easy to add. By using TOPS20 and the standard software packages that usually go with it, such as BASIC + 2, you'd have plenty of system for the hobbyist to work with.

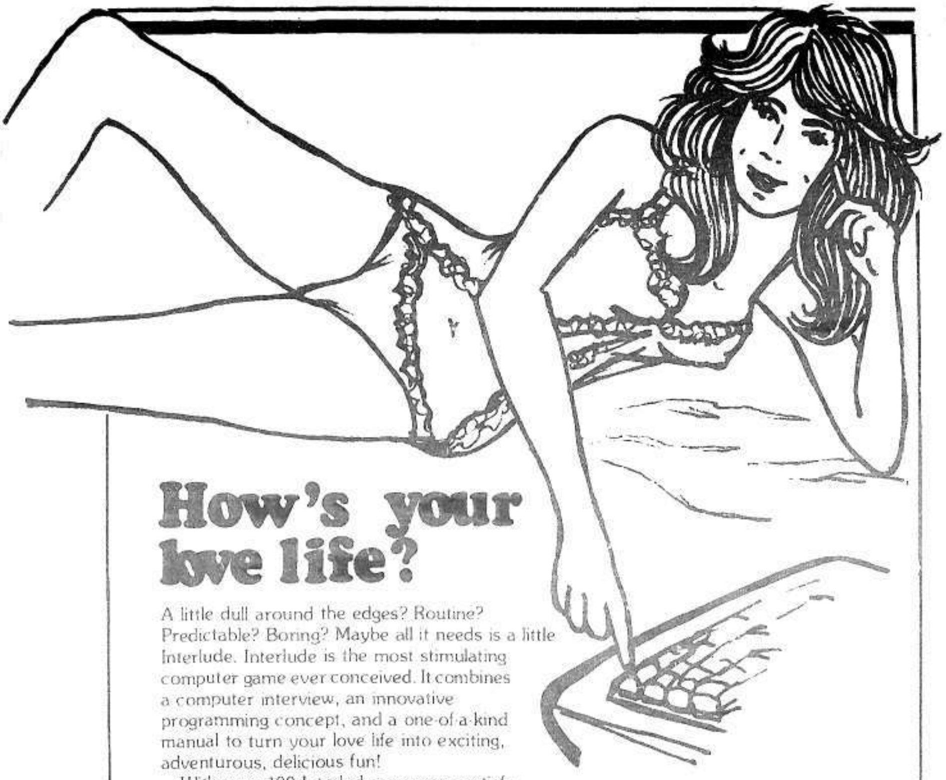
The next step is for the hobbiest to develop ideas which others will put to more practical use. Electronic mail, which includes not only sending messages but the sharing of data itself, will create its own self-satisfying needs the same way the telephone created its needs and then satisfied them. It starts out being something you do for fun, then it becomes something you rely on. Before you know it everybody has to have it. That's how the telephone became a necessity and how I think electronic mail will become a necessity.

Even my mother needs to send and receive mail, just like she must make and receive phone calls. If electronic mail becomes a common method of communication, she is now faced with a new need. Paid computer access satisfies that need. So, it's off to the computer store to buy an account, a video terminal, and a modem - the very stuff I couldn't talk her into.

To some, paid computer access seems to be the goose that lays golden eggs. If it is the perverbial goose, they want to be sure they've got their bets down on him. I prefer not to bet on any goose. All geese, including those who MIGHT lay golden eggs, ALWAYS lay goose eggs.

The 80-Users of Houston Club meets the first Wednesday of each month, at the Bellaire Chamber of Commerce Building, 6900 S. Rice in Bellaire, Texas at 7:30 pm. For more information, call Ben Taylor at (713) 664-5823.


LEVEL II
BASIC
Refere
Manu



How's your love life?

A little dull around the edges? Routine? Predictable? Boring? Maybe all it needs is a little Interlude. Interlude is the most stimulating computer game ever conceived. It combines a computer interview, an innovative programming concept, and a one-of-a-kind manual to turn your love life into exciting, adventurous, delicious fun!

With over 100 Interludes, you can satisfy all levels of interest and desire. Each Interlude is fully described in the manual, and the more elaborate ones are detailed with regard to settings, props, and mood-enhancing techniques. But we've saved a few super Interludes for that very special time when your interview indicates you're ready! At that time, you will be introduced to one of several Interludes held secret within the computer. (When you learn secret Interlude #99, your love life may never be the same again!) Interlude can give you experiences you'll never forget. Are you ready for it?

For the TRS-80(Level II, 16K)

$14.95 for cassette
$17.95 for diskette

Interlude™

The Ultimate Experience.

The Software Exchange

6 SOUTH ST., MILFORD, N.H. 03055 (603) 673-5144

GHOST TOWN

by Scott Adams

Scott's long-awaited 9th adventure is here! Thirteen treasures are hidden somewhere in the GHOST TOWN. You'll need more resourcefulness than ever to find them before sunset. Be ready for a town full of surprises and some REAL ghosts! Bring along a sense of humor, but be alert for danger as well. If you live so long, you MAY learn the answer to the tantalizing question:

"What goes on
in the saloon
after dark?"

Cassette $14.95

LEVEL II 16K

The Software Exchange
TSE 6 SOUTH ST., MILFORD, N.H. 03055 (603) 673-5144

A SECOND LOOK AT MICRONET and SOURCE IS NOW MNET-80 TOO

by Richard Taylor

This month's column is actually a combination of two related articles written to discuss MICRONET. One was a close look at MICRONET's many excellent features, and the other the story behind SOURCE-80's joining MICRONET. Together they give a good look at the system's offerings and capabilities.

I have mentioned in earlier articles that I was more familiar with THE SOURCE, having spent more time on it. Of late, I have been spending a lot of very rewarding time on MICRONET, and am pleased to be able to describe and discuss just a few of the superb features of this system.

MICRONET starts off with a big advantage in the cost department. The initial fee for joining is only $9.00 compared with THE SOURCE's $100.00. The first $5.00 of your $9.00 pays for your first hour on the system, which brings your initial investment down to only $4.00. There is no minimum monthly charge if you do not use the system and, as I said in a previous article, there is really no reason not to belong to MICRONET. (TSE includes a MICRONET membership with any SOURCE membership it sells).

You will not find UPI on MICRONET as you will not find most of the data bases offered by THE SOURCE. What you will find is a highly sophisticated time-sharing system with very fast response time and an average "up" time of 96% for the year. People who argue that MICRONET is more expensive than THE SOURCE ($5 vs. $2.75 per hour) should know that, as of this writing (April 1980), response time on MICRONET is at least twice as fast as THE SOURCE. Before I start detailing features on MICRONET I should digress a bit and fill you in on some background.

SOURCE-80

SOURCE-80 is a TRS-80 users group which I run on THE SOURCE.

We produce a huge volume of articles, original software, patches, etc. each week and it's my job to edit and send this information out to the entire group, using THE SOURCE's Mail System. Because of some problems that were occurring on THE SOURCE, I thought it might be a good idea to have a backup plan in case we were unable to communicate. I knew that MICRONET offered at least two things which could be very useful to the group:

1. 128K of free storage for each account;
2. The ability to make files available to other users without giving away your password.

It was my idea originally to establish a data base on MICRONET of our past material which members could access as they needed. The base, consisting of the group's communications, articles, programs, etc. now occupies more than 600K and is running very well. It was also my idea to establish a means of communication among group members on MICRONET. I realized that people using the data base were accessing text files and I thought it would be possible for people to run original programs from my file area. I was unsure how local disk I/O would work while users were running my software so I began to experiment. This led to CBBS.CMD, a bulletin board, which is, in effect, the first national TRS-80 computer bulletin board system. At this point, I should add that most SOURCE-80 members were already subscribers of MICRONET and those that were not decided to subscribe. Some members live in areas not served by a local MICRONET phone number and have to go through TYMNET to access the system. At the same time that SOURCE-80 was expanding to MICRONET, MICRONET announced a reduction in the cost of the TYMNET surcharge from $4 to $2 per hour. Now it was possible for SOURCE-80 members to share information and communicate on MICRONET as well as on THE SOURCE. Once the members started using MICRONET it quickly became obvious to all that MICRONET would be far more than a backup system. The speed and versatility were a joy to behold. I wrote the bulletin board program using MICRONET's COMPUSERVE EXTENDED BASIC. This is a powerful BASIC and compiles for excellent speed of execution. My bulletin board system runs with the same remarkable speed and reliability as most of the features on MICRONET. This is attested to by the fact that during a typical weekend there are over 200 accesses to the board. I kept it similar in format to a FORUM-80 so that people would feel comfortable using it for the first time. To access the board while on MICRONET all you do is type: IRUN CBBS.CMD[70161,101] (more on those brackets later). You will then be running an image file (compiled BASIC) of my program and using local disk I/O in my personal file area. Didn't I tell you this system was sophisticated? The idea of several people running the program at the same time, and accessing the disk files at the same time, was new to me and led to a week of rocky running. With help from the MICRONET staff (Wizards) and a manual, I soon got the board running smoothly. It now recognizes members when they run it, auto-logs them on, and gives them access to the data base.

ST80 AND MICRONET

I use ST80D and ST80III for all of my file uploading and downloading. This can be a precarious business on THE SOURCE as the speed controls have to be adjusted constantly. Lost characters are a common occurrence and in the case of program listings, one of the slowest speeds must be selected for the best chance of getting the file into the system error-free. MICRONET had a surprise waiting in this area, also. ST80D and ST80III use control "S" and control "Q" for handshaking which works flawlessly with MICRONET. A speed of 0 (the fastest) can be selected for uploading files to MICRONET and MICRONET will stop and start ST80 as it needs to. It's a real kick to watch and, better than that, it assures you of error-free transmissions to the point where you can leave the room to pour a cup of coffee knowing that everything is uploading to MICRONET just as it existed in your disk file.

My Bulletin Board has been a big success and more of them are springing up. I am customizing the software for various companies and there is already one running for Ham Radio users. Type: IRUN HAMNET.CMD<70000,24> to access the board. H11 and PDP11 users also have one of my boards running which can be accessed by typing IRUN MNET11.CMD<70110,403>. The characters surrounding the account numbers should be square brackets. I have deliberately left them as you see them (less than and greater than signs) because that is the way they are displayed with most terminal software. With ST80, as with most terminal programs, the brackets surrounding the account numbers are entered as control 8 and control 9. The MICRONET EXECUTIVE (a free terminal program offered by MICRONET) is the only terminal program I have ever seen which depicts these brackets graphically as square brackets.

DOCUMENTATION

The documentation which MICRONET sends you is very good. Their free BASIC handbook was enough to get my entire Bulletin Board program translated from Microsoft BASIC into Compuserve BASIC. The big differences were in the disk file I/O which I later found out are identical to DEC10 BASIC. Such features as the setting of protection levels, using the text editor, intricate file manipulation procedures, setting terminal parameters, writing programs on the system, and more are all discussed in detail in the documentation sent to you when you join. Additional manuals can be purchased for FORTRAN, AID, APL, BASIC, BLIS10, FOCAL, MACRO, and SNOBOL, among others. There are many text editors, a word processor, utilities of all kinds, financial programs and, of course, games. Serious programmers will be very pleased with what they can accomplish on this system.

USING THE TEXT EDITOR

A lot of what you do on MICRONET involves a text editor called FILGE. Whether editing one of your own text files, or sending a message to a friend on the MICRONET BULLETIN BOARD SYSTEM (not to be confused with mine), you will find yourself using FILGE. FILGE is the main text editor found on MICRONET. If you never do any editing, you can go through life knowing only one FILGE command ("/EX"). That's the command you type when you have finished entering your text and wish it to be saved. All FILGE commands begin with "/" at the start of a line. You can delete, read, move the line pointer, locate text, global search/replace, etc.. This may all seem a bit awkward for TRS-80 users who are used to using an on-screen character-oriented editor such as

ELECTRIC PENCIL or SCRIPSIT, but after a little practice you will find yourself "filging" everything in sight. Some of the FILGE commands will be discussed in the next two sections dealing with BULLET (the electronic mail system) and creating and transmitting files.

ELECTRONIC MAIL

The electronic mail system on MICRONET (BULLET) is not nearly as sophisticated as the mail system found on THE SOURCE. I am told, however, that a new mail system is soon to be unveiled on MICRONET and by the time this article appears, it may already be a reality. To read or send mail, type R BULLET. This will transfer you to another system and log you off of the original host. There are private messages (user to user) and public messages under the categories of NOTICE, SALE, and WANTED. To see the NOTICE messages, just type VIEW NOTICE. Each heading will be displayed with the choice of "Read Y,X,C/R". Entering "Y" will type the text of the message, and "C/R[ENTER]" will skip over the text and move on to the next heading. Using the "X" option will break you out of the VIEW mode and bring you back to the BULLET Prompt(*). VIEW USER will show you any private mail which is waiting for you. Sending a letter is a bit more complicated. From the "*" type COMPOSE. This will create a working file (FILGE!) and you just start typing. Please be aware that just like THE SOURCE mail system, you are not running ELECTRIC PENCIL! There is a line length limit in effect and the best practice is to hit [ENTER] at the end of each video line. Failure to do this will result in some very strange messages. A typical message which I receive from new users goes something like this:

"Dear Richard,
You must help me with a very serious problem. I'm at my wits end and if I don't get the answer soon I'll throw my computer out the window. How do"

And that's the end of the message—122-140 characters (depending on how you signed on) and the system truncates everything from that point until it finally gets a C/R [ENTER].

Once you have composed your message, start the next line with "/EX". This ends the FILGE section and saves the file. Now the file is waiting to be routed somewhere. If you were sending this letter to me you would type (at the "*") POST USER 70161,101 KEYWORD. Where KEYWORD is the subject. Don't include a space as in "A SURPRISE" because the subject will be taken as "A" (I get a lot of those also). You must have a Keyword or you will get an error message until you type the line correctly. The letter has now been sent to me and the file is still there if you wish to send the same letter to anyone else. Users get a message when they log-on telling them of any personal mail. If you want the message to be public just use the word NOTICE, SALE, or WANTED in place of USER and omit the account number.

There are two ways to get out of BULLET. Typing OFF will sign you off the entire system. Typing EXIT will get you out of BULLET and return you to the regular MICRONET host.

TRANSMITTING FILES

There are several ways to create files in your disk file area. One of the easiest is through our old friend FILGE. When you have your file loaded into the ST80 buffer, type FIL FILENAME.XXX. FILENAME can be up to 6 ASCII characters with the first being a letter. The XXX is any extension such as TXT for text, BAS for BASIC, XBA for Extended BASIC, XF4 for Extended Fortran and so on. FILGE does not care whether or not your file has line numbers. After FILGE tells you that it has created a new file you may begin entering your text. As I said before, ST80D and ST80III can be used at full speed with no worry. The whole thing will crash however, if you have lines which exceed 140 characters in length. Once the file has finished uploading, do any editing you need to do and type "/EX" at the start of the next line. This ends the FILGE session and saves the file.

If the file you are creating is a BASIC program you may want to load it into BASIC rather than FILGE. To do this, type NEW FILENAME.XBA (or .BAS). The extension tells MICRONET which language the program is written in. In this mode, each line must begin with a line number. Typing SAVE on a new line will save the contents of your file to disk. This is just the same as being in BASIC on a TRS-80 and you may run what you have entered at any point. When you type RUN the program is compiled and any errors will be pointed out to you at this time. If the errors are serious enough, the program will not run until you correct them. Typing COM FILENAME compiles the program and creates a core image file called FILENAME.SAV. This file can be saved through the command SSA FILENAME.SAV and the file can be executed at any time with the command IRUN FILENAME.SAV.

EDITING YOUR FILES

Any ASCII file can be called into your workspace for editing. If the file has line numbers it is best to type: OLD FILENAME.XXX. This command places a copy of the file in your workspace while keeping a backup of the original file in case things go badly during your editing. Individual lines can then be addressed by line number and retyping a line will replace the original. No changes are made to the original file until you type REPLACE. This replaces the original file with any changes you have made while editing. Using this system allows you to RUN the program to check out your changes before making the changes permanent.

Typing FILGE FILENAME (where FILENAME is the file to be edited) also makes a backup but editing is limited in most cases to the line at which the line pointer is currently positioned. The command "/EX" saves the changes to disk. The editing commands are many and varied for each of these systems and all commands are well-documented in the MICRONET Manual.

SHARING FILES

If you wish other users to be able to read your files you must change the protection levels of the file and your directory. Typing PRO FILENAME(2) sets the protection level at (2) which allows other users to read and copy the file. If the file is a program you could set the protection level to (8) which allows other users to execute only. Along with these protections you must have your directory protection set to allow users to access your files. This is done through the command PRO#(nnn). This time those brackets are actually GREATER THAN and LESS THAN signs. A protection level of (744) allows the reading of files while still keeping your directory unreadable. (755) allows reading both files and your directory while (777) will even allow other users to create files in your area. I keep my directory protected from reading while enabling access to the data base files (if you know the exact name of the file). Group members are given access to a special file which contains a detailed directory of the files available in the database.

MNET-80 IS BORN

Micronet proved to be such a fine system that now in addition to SOURCE-80 we also have MNET-80. In addition to all the SOURCE-80 members, we now have quite a few members who only belong to MICRONET. The bulletin board (CBBS.CMD) has made communication very easy and the database system allows new members to choose anything they wish from past material. Many TRS-80 users have MICRONET accounts but they have not been using them. I hope this article will give you reason to start. While you're there just type IRUN CBBS.CMD[70161,101] and join the first "Real Time TRS-80 Users Group".

Put IRV on your programming staff!

Comes with manual

Input shorthand — one keystroke can enter a whole line! The feature you liked in Level III BASIC and T-Short, now with all keys user programmable! You can even turn your cassette player on or off with a single keystroke.

Relocate a line by simply editing the line number — renumber lines individually!

Video Editor allows cursor oriented editing of your programs. Just move the flashing cursor to the line to be edited and type the corrections, insert, or delete!

IRV is faster than Level II BASIC! More powerful than a speed typist! Able to leap ten subroutines at a single bound!

Machine language tape $24.95 **On Disk $29.95**

The Software Exchange

6 SOUTH ST., MILFORD, NH 03055 673-6144

GARON'S GOODIES

by James Garon

PROGRAM PORTABILITY

Writing a nifty utility program in Level II? Want disk users to be able to RUN your efforts? Perhaps you've developed a super renumber routine or maybe a SQUISH-type blank remover. If you make use of the Start of Program address of 17129 (42E9 hex), your program may work like a charm in Level II, but it doesn't stand a chance in Disk BASIC.

Not only do programs running in Disk BASIC NOT start at 17129, they may not even start at the same place from one day to the next! It will depend upon which DOS is in control and upon how many files are open. Wouldn't it be nice if there was some way to find out where a BASIC program was hiding? There is!

Wherever your old utility program refers to the address 17129, such as:

```
100 LET E=17129
```

replace this with:

```
100 LET E=PEEK(16548) +
    256*PEEK(16549)
```

Locations 16548 and 16549 contain an address which always points to the beginning of program text—thus both Disk and Level II users can RUN your program with equal ease.

While we're on the subject of making programs portable between Level II and Disk BASIC, let's discuss the way to initialize those USR calls. In Level II, we poke the starting address into locations 16526 and 16527. Assuming, for instance, that your USR routine begins at location 7000 (hex), you might use the following line in Level II:

```
200 POKE 16526,0:POKE 16527,112
```

as explained on page 8/8 of your Level II manual. (By the way, there is an error in the explanation on that page: you should replace both references to the number 208 with 125.) But Disk BASIC has a special command for setting up a USR call; if we didn't care if our program worked in Level II, we could use:

```
200 DEFUSR=30000
```

or:

```
200 DEFUSR=&H7000
```

If only there was some way for the

program to discover if it is currently running in Level II or in Disk BASIC...Again, we are in luck! Location 16396 is just waiting to reveal this information. Try this:

```
200 IF PEEK(16396)=201 THEN POKE
    16526,0:POKE 16527,112 ELSE
    DEFUSR=30000
```

This works because Disk BASIC places a 195 into 16396 (a machine language Jump instruction) while Level II leaves a 201 there (a Z-80 Return). This location is checked whenever the BREAK key is pressed. Disk BASIC has to do some extra checking since BREAK may awaken the terrifying DEBUG from its fitful slumbers.

How's your love life?

A little dull around the edges? Routine? Predictable? Boring? Maybe all it needs is a little Interlude. Interlude is the most stimulating computer game ever conceived. It combines a computer interview, an innovative programming concept, and a one-of-a-kind manual to turn your love life into exciting, adventurous, delicious fun!

With over 100 Interludes, you can satisfy all levels of interest and desire. Each Interlude is fully described in the manual, and the more elaborate ones are detailed with regard to settings, props, and mood-enhancing techniques. But we've saved a few super Interludes for that very special time when your interview indicates you're ready! At that time, you will be introduced to one of several Interludes held secret within the computer. (When you learn secret Interlude #99, your love life may never be the same again!) Interlude can give you experiences you'll never forget. Are you ready for it?

For the TRS-80(Level II, 16K)**

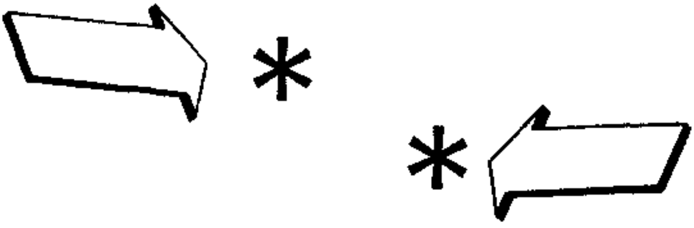
$17.95 for diskette

Interlude™

The Ultimate Experience.

The Software Exchange

6 SOUTH ST., MILFORD, N.H. 03055 (603) 673-5144



SPLAT — AN INTRODUCTION TO TINY COMP

by David Bohlke

Have you ever wanted to create games or routines with the speed of machine language and the ease of programming in BASIC? This is now possible with the use of the Level II BASIC compiler program TINY COMP. SPLAT is presented in this article as a sample program to illustrate some of the commands which can be converted to machine language by TINY COMP.

SPLAT is very simple in concept. An asterisk will appear on the screen, and the player must use the four arrow keys to move the block cursor and intercept the target. The sequence is timed and scored, and there are ten speed levels from which to select.

All the program lines, once compiled, will execute exactly as in BASIC — except they will run with machine language speed. The only exception is the INKEY$ function, as illustrated in line number 22 of the program listing. Q$=INKEY$ will return the ASC value of the key pressed in the variable Q. It does not return a String variable. The second statement in line 22 (IFQ$=""THEN22ELSEQ=ASC(Q$)) is added so that the program will RUN under BASIC with the same result as the compiled code. This second statement is NOT compiled by TINY COMP.

Since the compiled code executes the same as the BASIC code, it is fairly easy, using BASIC, to write and debug the programs before they are compiled. The compiler, source BASIC code, and the object code buffer all reside in a 16K Level II machine at the same time. There is no need to switch tapes back and forth every time you want to adjust the program listing. The BASIC code for SPLAT is a little over 1000 bytes; and the compiled code is 1142 bytes. In a 16K machine there is sufficient memory to compile a program over 6000 bytes in length.

Although not included in SPLAT, TINY COMP (tape version) can also compile the PEEK and POKE commands (as well as * and /). These can be used to build and access data tables and arrays. Also, the PEEK command can be used for scanning the keyboard for input.

To get the SPLAT program up and running, you must first CLOAD the TINY COMP compiler after answering memory size? with 26000. Next, key in the SPLAT listing. A line listing description is included so you can identify the SPLAT routines. If you would like a comparison of the BASIC speed, you can type RUN to execute SPLAT under Level II BASIC. In addition, when you are writing a program from scratch, you will be able to check and adjust program flow under BASIC execution. Remember, your BASIC source program must use line numbers 1-799, and it MUST contain an END statement.

When you are ready to compile SPLAT, type RUN1000. The compiler will display each line number of the current line being compiled, as well as the current location in the object code buffer, and the decimal equivalents of the compiled object code. TINY COMP can compile only the statements illustrated in the statement set table. Similar statements with the same key words (PRINT, IF, etc) which cannot be compiled will produce an ERROR LINE # message. Lines with other key words, like DEFINT A-Z, will be skipped by the compiler. Compile time is about 1000 bytes every 3-4 minutes.

After your program is compiled, just press ENTER when the prompt appears to execute the machine language code. If your program has a logical end, control will return to BASIC after execution is completed. To run the program additional times, type RUN1300. When the program has an endless loop, as in SPLAT, you will have to press the RESET button to return to BASIC.

The BASIC source program SPLAT, along with TINY COMP, can be CSAVEd for future use. The object code cannot be SAVEd in the same fashion, so it will be necessary to recompile (RUN1000) SPLAT on power-up.

The disk version of TINY COMP functions in a fashion essentially similar to the tape version. Some rearrangement of object code routines was necessary, since source code and compiler will not fit into the 6K left by disk BASIC (in a 16K machine).

Even though the writing of a BASIC compiler began as an experiment for me, I believe it has developed into a useful configuration. SPLAT is presented as a demonstration of the high speed capabilities of the compiled code in games. A more complex source program, 3D TIC TAC TOE, is included with the tape version of the TINY COMP compiler. Whether you believe a BASIC compiler can be useful for gaming, or just experimentation, I hope TINY COMP can be a learning experience for you as it has been for me.

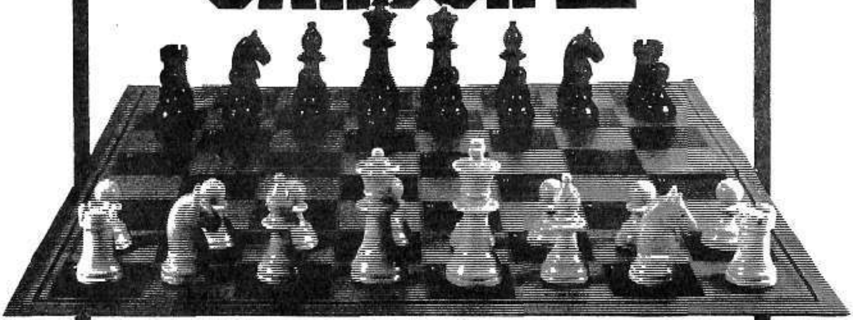
TINY COMP Statement Set

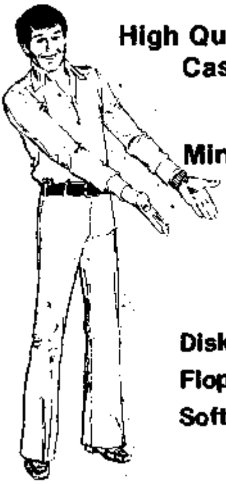
In the following statements — A,B,C, represent legal variable names (A-Z). The code ccc is a positive integer constant, and nnn represents a legal line number (1-799).

DISPLAY:	PRINT@A,B			
	PRINT@A,"STRING"			
	PRINT@A,CHR$(B);	semicolon is optional		
INPUT:	A$=INKEY$			
	A=PEEK(B)			
	POKE A,B			
ASSIGNMENT:	A=ccc	A=B	A=RND(B)	
	A=B+C	A=B-C	A=B*C	A=B/C
BRANCH:	GOTOnnn			
	GOSUBnnn	RETURN		
	IF A=B THEN nnn		< > also supported	
OTHERS:	END			
	REM			
	CLS			

SPLAT Line Listing Summary

10-30	Initialization
110-250	Main game loop
110	Print splatter, delay
120-124	Check for hit
130	Loop to continue
200-250	Increment points, reset screen
600-602	Set new target
650-654	White screen
700-737	Adjust cursor
740-746	Delay loop, check for end of game
750-752	Print splatter

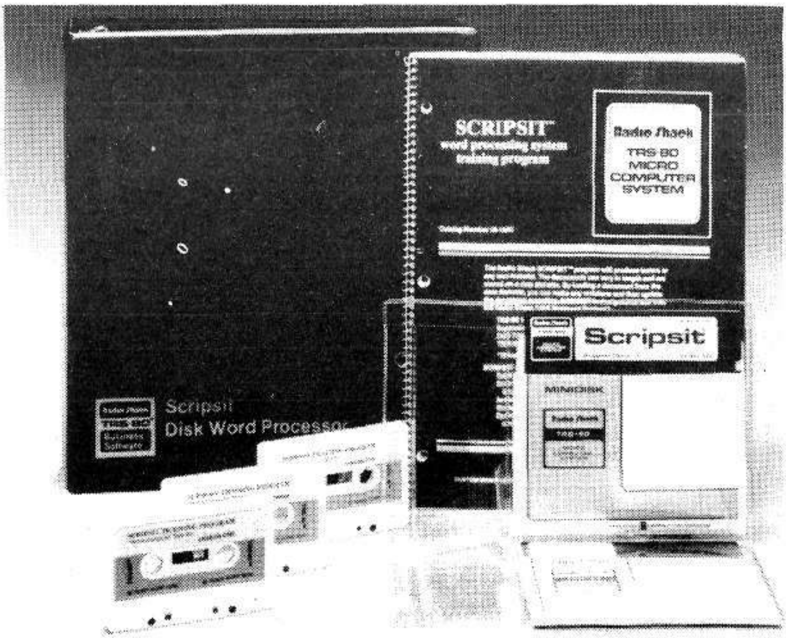




LEARNING LEVEL II
LEARNING LEVEL II
By David A. Lien

SCRIPSIT
word processing system
training program
Radio Shack
TRS-80
MICRO
COMPUTER
SYSTEM
Scripsit
MINIDISK
Scripsit
Disk Word Processor

"SCRIPSIT" FROM RADIO SHACK



IS IT TIME TO UNPLUG YOUR ELECTRIC PENCIL?

by Richard Taylor and Mark Kirmayer

SCRIPSIT, from Radio Shack, is a character-oriented word-processor written in machine language. It represents a giant step forward in the quality of Radio Shack software. It offers many features not found in the most popular TRS-80 word-processing program (THE ELECTRIC PENCIL), and goes one step beyond with its extended features for print formatting. SCRIPSIT offers almost limitless print formatting options, tremendous versatility in the entering and editing of text, complete and detailed documentation, and is available in cassette and disk versions, both of which include an upper only and an upper/lower case copy of the program. The program loads any ASCII file (which includes PENCIL files) and is compatible with the lower case modification described in the ELECTRIC PENCIL manual. Radio Shack is now offering its own lower case modification for the price of $100, installed.

Anyone familiar with ELECTRIC PENCIL will be pleased to know that SCRIPSIT does not require slowing down the typist as he or she approaches the end of the video line to avoid lost characters as the text is adjusted. Touch typists can whiz ahead at their own speed with no need to look at the screen for verification.

Formatting and Editing

The print formatting commands are extensive and sophisticated. In addition to the standard controls over PAGE LENGTH, LINE LENGTH, LINE SPACING, PAGE NUMBERING, and RIGHT JUSTIFICATION, SCRIPSIT adds commands such as VERTICAL CENTERING, HORIZONTAL CENTERING, PAUSE BETWEEN PAGES, SPACING BETWEEN PARAGRAPHS, HYPHENATION, FLUSH RIGHT PRINTING, HEADERS, FOOTERS, and WIDOW SUPPRESS. The truly fantastic thing about all these commands is that any printing parameter may be changed anywhere in the document. Lines beginning with a "Greater Than" sign (>) are called FORMAT LINES and are non-printing. These lines are placed within the text at the point you wish to change the formatting. The lines are interpreted while the document is being printed and the format of the printout is changed accordingly. The lines remain as part of the text file and are saved to tape or disk. In addition, BOUNDARY MARKERS allow you to define lines, paragraphs/and pages thereby giving you even more control over the finished product. Individual HEADERS and FOOTERS may be defined for even and odd pages or turned on or off in the FORMAT LINES. People who do not use continuous-feed paper will be pleased to see the PAUSE BETWEEN PAGES option. This PAUSE option, used in conjunction with a FLUSH RIGHT HEADER on even pages and a FLUSH LEFT HEADER on odd pages, combined with a CENTERED FOOTER consisting of the PAGE NUMBER, allow you to print on both sides of the paper and end up with a finished product in book-style format. This may sound confusing but it begins to give you an idea of what this program can do. Additionally, the program can also produce a perfectly centered (both vertically and horizontally) title page.

PENCIL users will find the text-editing commands more difficult to use at first, but they will also find more versatility. Instead of different one letter commands for each function, SCRIPSIT uses "Modes" such as DELETE, INSERT, EXCHANGE, BLOCK, and REPEAT. The "@" key is the CONTROL KEY and is used for all commands. Other keys are used to define WORD, LINE, PARAGRAPH, START, END, and BLOCK. The first command puts you in an editing "Mode" and the second command defines the amount of text you wish to edit. For instance, typing CONTROL D puts you in DELETE MODE (and you are told so at the bottom of the screen). While in DELETE MODE, you may type CONTROL X to delete the sentence or CONTROL Z to delete a word. The same is true for the INSERT MODE and, to some degree, for the other "modes" listed above. The versatility comes with the BLOCK and EXCHANGE MODES. Any amount of text may be marked and named as a BLOCK for future manipulation. The EXCHANGE MODE easily allows you to switch two words, paragraphs, or BLOCKS of text.

GLOBAL SEARCH/REPLACE/DELETE commands allow you to find a specified string in the text.

This can be used to find a section of text and/or to delete it. The command can also be used to replace the specified string with another string of any length. The REPEAT MODE allows you, as does PENCIL, to enter a number sufficiently large to execute the command automatically up to 255 times.

Self Teaching Cassettes

SCRIPSIT comes packaged with three audio cassettes and a set of labels for the keyboard. The audio cassettes take the listener through every feature of the program with agonizing detail and were obviously designed for a person new to the TRS-80. Still they are well-done and, once through, you will be totally familiar with all the various functions. Sample text files are also included on the program cassette or disk for use with the course. The documentation is thorough and even includes a section for programmers to help them access the SCRIPSIT-generated files in their own programs.

Disk I/O is greatly improved over PENCIL and disk space is used more efficiently. LOADING and SAVING time are much shorter than in PENCIL. As mentioned earlier, any ASCII file can be loaded into SCRIPSIT, and this includes BASIC programs. Programs saved in ASCII format can be loaded into SCRIPSIT and edited. The number of continuous characters without a space is of no importance here as it is with PENCIL.

Full cursor control via the arrow keys is provided as well as a fantastic feature which lets you define the WINDOW parameters of the video display. Any line length from 1 to 132 can be represented on the screen. This is accomplished by having the text scroll sideways across the screen! Think of your video as a window passing across the 80 character (e.g.) length line. If you have ever tried to write a chart for an eighty column printer with PENCIL, you will realize the importance of this feature. This feature also allows you to see how your text will appear under different line lengths. If charts or reports are your aim, SCRIPSIT lets you preset TABS which are represented at the bottom of the screen for your reference.

SCRIPSIT also allows you several questions about the document and the screen status which you can ask at any point. You may ask for the amount of remaining memory, the current screen width, the length of the document in characters, what length the PARAGRAPH INDENT is set for, the NAME of the document, and which line the cursor is currently on.

Disk users might like the fact that the cursor position has nothing to do with the amount of text written to disk during a SAVE, as the entire document is written to disk. Every PENCIL user has, at some time, saved his composed text without realizing that the cursor was not at the beginning of the file. By the time he realizes his mistake it is usually too late. The disk version also remembers the name of the document loaded into memory. To save the new version of the document, a simple "S" will update on disk using the same name under which it was loaded. Loading in a new file deletes the present document unless the CHAIN switch is used, in which case the new file is appended to the one currently in memory.

Electric Pencil

By this time you must think that I am recommending a bulk-eraser for your ELECTRIC PENCIL. Not so! SCRIPSIT is a fantastic program but it will never totally replace

ELECTRIC PENCIL. The one big drawback to SCRIPSIT is its lack of a Disk Directory function. There is no way to see what is on the disk and there is no way to KILL a file. I use PENCIL all the time to edit and MERGE files generated by ST80D (a terminal program) for my Users Group on THE SOURCE. I am constantly editing, merging, and then killing files to make room on the disk. None of this is possible with SCRIPSIT. Another advantage PENCIL has over SCRIPSIT is its easy ability to print only as many lines of text as you wish. I use PENCIL to locate and print addresses from a large mailing list. By telling PENCIL "P3", it will then print only three lines, starting with the current cursor position. There is no comparable command in SCRIPSIT. It can be done, but it would involve changing "Markers" for each address you wished to print.

SCRIPSIT for cassette at $79.95 undersells PENCIL by $20.00 and SCRIPSIT for disk at $99.95 undersells PENCIL by $50.00. If you are having the Radio Shack Lowercase Mod installed, be aware that the cassette PENCIL will not work with this mod because of the lack of a control key. Disk users with the Radio Shack Lowercase Mod can use PENCIL only on VTOS 3.0. VTOS 3.0 (and 3.1) use the Shift plus Down-Arrow as a control key, making it possible to use PENCIL with the Radio Shack Lowercase Mod.

Our final note—SCRIPSIT does not work with NEWDOS. APPARAT has the short patch necessary to make SCRIPSIT conform with the NEWDOS method of handling the End-of-File Marker. Rather than giving you the patch here, APPARAT has requested that you write to them so that they can send you the new up-to-date listing of NEWDOS patches. Their address is: APPARAT INC., 6000 E. Evans Ave., Bldg #2, Denver, Colorado 80222. They will send the list of ZAPs to anyone who requests it.

"SHE'LL HAVE TO CALL YOU BACK. SHE'S DOING ANOTHER CHANGE REQUEST NOW"

TRS-80 PROGRAM STORAGE AND SOME USEFUL COROLLARIES

by Jeff Meyers

Recent investigations into the TRS-80's method of program storage, aided by the memory map in the user's manual, have led to some short programs which I find useful. These experiments originated in hopes of finding a method of resetting the DATA pointer (a memory location where the location of the current DATA line being READ is stored) to any particular desired line. I have worked with BASICs in which there were statements such as RESTORE 222 which do this job, and occasionally I feel the need for this feature.

By means of a memory monitoring program I've written, I started at decimal 17128 (which is always set to zero) and found the BASIC program beginning in the next byte according to the format shown in Chart 1. The first two bytes were the Least Significant Byte (LSB) and the Most Significant Byte (MSB) of the address of the next BASIC line. When these "pointers" are both zero, this signifies the end of the BASIC program. Following these pointers are the LSB and MSB of the BASIC line number followed by the line of the program in a special code, with the end marked with a zero.

Chart 2 shows a list of the keyword codes as utilized in BASIC program storage. This set of words can be found in ASCII form, extending from byte 5712 to byte 6175. (The first letter of each word is ASCII plus 128.) When these words appear in a BASIC program, they are coded as a single byte.

One more technicality needs to be understood before experimenting with program storage: line numbers following GOTOs, GOSUBs, THENs, and ELSEs (which will hereafter be called referenced lines) are not stored in the LSB-MSB format, but as single digits in ASCII form. For example, in Chart 1, note that the line numbers following the GOSUB and the GOTO require three bytes each, one per digit.

No doubt there are reasons, unknown to novices like myself, for this method of coding GOTOs, etc., but there are BASICs in which the two byte form is used, and that form makes tasks such as renumbering a lot easier!

But back to my original goal — an adjustable RESTORE capability. By purely experimental means, I located the spot at which the computer "remembers" the line currently being read. The address of the byte ahead of that DATA line is stored at 16639 and 16640. With that information, I was able to write the BASIC program (Example 1) below which is called as a subroutine with Q being sent as the line number to which you wish to reset the DATA pointer. The next READ statement will begin at the first piece of data in that line.

This little program adds flexibility in certain programs but it is not the world's fastest subroutine. This prompted me to improve it by writing a machine language version. It is not necessary to understand the inner workings of this, but the appendix contains an assembly language listing of it, with comments. Example 2 stores this routine into the memory locations occupied by string variable Q$. This much faster-running method of resetting the DATA pointer is called simply by USR(XXX) where XXX is the desired line number. The program needs to be run only once.

As is always the case in programming (as in life), the answer to the question above led to a dozen new questions. The general drift of these new ideas was "what else can be done now that I know how programs are stored?" Needless to say, renumbering came to mind.

The program in Example 3 below is painfully slow and undoubtedly unsophisticated but it will renumber a BASIC program based on two assumptions: 1) that there are no commas used in place of THENs, and 2) that all referenced line numbers are exactly four digits long. This second restriction removes the necessity of sliding the program to make room for line numbers in GOTOs, etc., when the new referenced line has more digits than the old one (which would require that the variables at the program's end be moved also). It was written to be used from cassette tape at the beginning of a project which might eventually need renumbering, with that program being written starting at 1000.

While its restrictiveness and slowness are drawbacks, it does (barely) satisfy the challenge I set for myself at the start of its development — to have it fit on the screen!

After accomplishing the crude renumberer, a simple corollary idea presented itself. Example 4 lists on the screen all of the line numbers which are referred to by GOTOs, GOSUBs, THENs and ELSEs. I find this useful when trying to compact a program such as a game typed from **SoftSide** in which the author didn't use colons and which takes a long time to load from cassette.

In this program, the 202, 141, 145, and 149 are the codes for THEN, GOTO, GOSUB, and ELSE respectively. Line 10040 can be altered to also check for commas (substituting for THENs) by adding a W=44 condition but if there are POKE statements or certain types of PRINTs, the printout might contain extraneous numbers.

Finally here is one more spin-off from my program storage experiments. While testing various bytes in the mid-16000's, I found that 16633 and 16634 point to the address of the first variable defined and that 16635 and 16636 point to the last. The variables are defined during the run of a program and occur in the order created. When "looking up" a value for a variable, the computer scans the whole list. This is why it's important to define much-used variables early if you are trying to speed up execution.

This list of variables is located at the end of the BASIC program and Example 5 uses this list to print a list of all non-array variables used during a program's run (plus variables 01,02, and 03 from Program 5). Jump to this routine from the end of a program.

The examples shown here may serve as starting points for investigations of your own. In my experience, there is a virtually limitless variety of ways to get into trouble when toying with program storage, but there is also a possibility of very useful results. Programs could theoretically be altered in midstream by careful POKEs into the program listing, DATA could be altered, and whole lines rewritten in the middle of a run! Good luck!

CHART 1: PROGRAM STORAGE FORMAT

Decimal Address	**Decimal Contents**	**Comments**
17128	0	never changes
17129	243	243+256*66=17139 (pointer)
17130	66	
17131	10	line number 10
17132	0	
17133	145	GOSUB code
17134	32	ASCII for space
17135	57	" " 9
17136	57	" " 9
17137	57	" " 9
17138	0	line end marker
17139	253	253+256*66=17149 (pointer)
17140	66	
17141	20	line number 20
17142	0	
17143	141	GOTO code
17144	32	ASCII for space
17145	49	" " 1
17146	49	" " 1
17147	49	" " 1
17148	0	line end marker
17149	(etc.)	(next pointer, next line number, next line, etc.)

The listing at the left represents the contents of memory assuming that the BASIC program begins

10 GOSUB 999
20 GOTO 111
etc.

CHART 2: PROGRAM STORAGE CODES FOR THE TRS-80 (DECIMAL)

128	END	154	DEFSNG	180	LIST	206	-	232	CVD
129	FOR	155	DEFDBL	181	LLIST	207	*	233	EOF
130	RESET	156	LINE	182	DELETE	208	/	234	LOC
131	SET	157	EDIT	183	AUTO	209	↑	235	LOF
132	CLS	158	ERROR	184	CLEAR	210	AND	236	MKI$
133	CMD	159	RESUME	185	CLOAD	211	OR	237	MKS$
134	RANDOM	160	OUT	186	CSAVE	212	>	238	MKD$
135	NEXT	161	ON	187	NEW	213	=	239	CINT
136	DATA	162	OPEN	188	TAB(	214	<	240	CSNG
137	INPUT	163	FIELD	189	TO	215	SNG	241	CDBL
138	DIM	164	GET	190	FN	216	INT	242	FIX
139	READ	165	PUT	191	USING	217	ABS	243	LEN
140	LET	166	CLOSE	192	VARPTR	218	FRE	244	STR$
141	GOTO	167	LOAD	193	USR	219	INP	245	VAL
142	RUN	168	MERGE	194	ERL	220	POS	246	ASC
143	IF	169	NAME	195	ERR	221	SQR	247	CHR$
144	RESTORE	170	KILL	196	STRING$	222	RND	248	LEFT$
145	GOSUB	171	LSET	197	INSTR	223	LOG	249	RIGHT$
146	RETURN	172	RSET	198	POINT	224	EXT	250	MID$
147	REM	173	SAVE	199	TIME$	225	COS		
148	STOP	174	SYSTEM	200	MEM	226	SIN		
149	ELSE	175	LPRINT	201	INKEY$	227	TAN		
150	TRON	176	DEF	202	THEN	228	ATN		
151	TROFF	177	POKE	203	NOT	229	PEEK		
152	DEFSTR	178	PRINT	204	STEP	230	CVI		
153	DEFINT	179	CONT	205	+	231	CVS		

PROGRAM 1; VARIABLE DATA RESTORE IN BASIC

```
600 DATA1,2,3
700 DATA4,5,6
800 DATA7,8,9
50000 'RESETS DATA POINTER TO LINE Q WHEN CALLED
50010 B=256:J=16548:M=PEEK(J)+256*PEEK(J+1)
50020 IFPEEK(M+2)+B*PEEK(M+3)<>Q THEN M=PEEK(M)+B*PEEK(M+1):GOTO
50020
50030 M=M-1:A=INT(M/B):POKE16639,M-B*A:POKE16640,A:RETURN
```

PROGRAM 2: VARIABLE DATA RESTORE WITH MACHINE LANGUAGE

```
10 'ROUTINE TO LET USR(XX) REESTABLISH DATA POINTER TO LINE XX
20 Q$=STRING$(38,1):Q=VARPTR(Q$):M=PEEK(Q+1)+256*PEEK(Q+2)
30 FORZ=MTOM+37:READX:POKEZ,X:NEXT
40 POKE16526,PEEK(Q+1):POKE16527,PEEK(Q+2)
50 DATA 205,127,10,68,77,33,233,66,94,35,86,35,126,185,32,15,35,
126,184,32,10,43,43,43,43,34,255,64,195,154,10,122,179,235,32,22
8,24,246
```

PROGRAM 3: BASIC RENUMBERING PROGRAM

```
1000 GOTO1010
1010 IFU=1THEN1020ELSE1030
1020 GOTO1000
1030 IFJ=0THENGOTO1030
1040 GOSUB1050
1050 RETURN
10000 DEFINTA-Z:INPUT"START,INCREMENT";S,I
10010 DIMA(500):C=1:Q=16548:M=PEEK(Q)+256*PEEK(Q+1):O=M:B=256
10020 J=PEEK(M)+B*PEEK(M+1):A(C)=PEEK(M+2)+B*PEEK(M+3)
10030 IFA(C)>9999THENM=O:GOTO10060ELSEQ=S+C*I-I:U=INT(Q/B)
10040 POKEM+3,U:POKEM+2,Q-B*U:C=C+1:M=J:GOTO10020
10060 J=PEEK(M)+B*PEEK(M+1):K=M+4
10080 W=PEEK(K):IFW=202ORW=141ORW=145ORW=149THEN11000
10090 K=K+1:IFK<JTHEN10080ELSEM=J:GOTO10060
11000 K=K+1:IFPEEK(K)=32THEN10080
11005 IFPEEK(K)<48ORPEEK(K)>57THEN10080
11010 Y=0:FORQ=0TO3:Y=10*Y+PEEK(K+Q)-48:NEXT
11020 FORX=1TOC-1:IFY=A(X)THEN11030ELSENEXT:END
11030 N=S+X*I-I:FORQ=3TO0STEP-1:POKEK+Q,48+N-10*INT(N/10):N=INT(
N/10):NEXT:GOTO10090
```

PROGRAM 4:
PRINTOUT OF ALL REFERENCED LINE NUMBERS

```
50000 'LISTS ALL REFERENCED LINE NUMBERS
50010 CLS:Q=16548:S=PEEK(Q)+256*PEEK(Q+1):TB=0:B=256
50020 T=PEEK(S)+B*PEEK(S+1):IFT=0THENENDELSEP=S+4
50040 W=PEEK(P):IFW=202ORW=141ORW=145ORW=149THEN50060
50050 P=P+1:IFP<TTHEN50040ELSES=T:GOTO50020
50060 P=P+1:W=PEEK(P):IFW=32THEN50060
50070 IFW<48ORW>57THENF=0:GOTO50040
50080 IFF=0THENTB=TB+8:IFTB>56THENTB=0:PRINT
50090 F=1:PRINTTAB(TB)CHR$(W);:GOTO50060
```

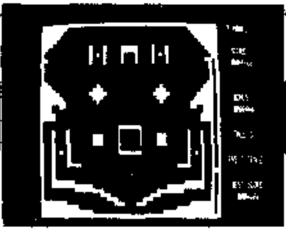
PROGRAM 5: VARIABLE LIST

```
 50000 'LISTS ALL DEFINED VARIABLES;JUMP HERE AT END
 50010 O1=PEEK(16633)+256*PEEK(16634):O2=PEEK(16635)+256*PEEK(166
36)
 50020 O3=PEEK(O1):PRINTCHR$(PEEK(O1+2));:IFPEEK(O1+1)>0THENPRINT
CHR$(PEEK(O1+1));
 50030 IFO3=3THENPRINT"$";
 50040 PRINT" ";:IFO1<>O2THENO1=O1+3+O3:GOTO50020
```

APPENDIX:

ASSEMBLY LANGUAGE LISTING FOR VARIABLE RESTORE

```
Decimal  Hex  Label   Mnemonic      Comment
  205    CD           CALL 0A7F     ;GET HL — OBJECT LINE NUMBER
  127    7F
   10    0A
   68    44           LD B,H        ;BC GETS OBJECT LINE NO.
   77    4D           LD C,L
   33    21           LD HL,42E9    ;HL POINTS TO FIRST BYTE
  233    E9                         ;OF PROGRAM STORAGE
   66    42
   94    5E   TOP     LD E,(HL)     ;DE GETS POINTER
   35    23           INC HL
   86    56           LD D,(HL)
   35    23           INC HL
  126    7E           LD A,(HL)     ;A GETS LSB OF LINE NO.
  185    B9           CP C          ;IF A ≠ C, MOVE ON
   32    20           JR NZ,CYCLE
   15    0F
   35    23           INC HL
  126    7E           LD A,(HL)     ;A GETS MSB OF LINE NO.
  184    B8           CP B          ;IF A ≠ B, MOVE ON
   32    20           JR NZ,CYCLE
   10    0A
   43    2B           DEC HL
   43    2B           DEC H L
   43    2B           DEC HL
   43    2B           DEC HL
   34    22           LD (40FF),HL  ;MAKE IT HAPPEN BY ALTERING
  255    FF                         ;DATA POINTER
   64    40
  195    C3   RETURN  JP 0A9A       ;RETURN,PASSING HL BACK...
  154    9A                         ;IF VALUE>0, LINE NO. WAS OK
   10    0A
  122    7A   CYCLE   LD A,D        ;"MOVE ON" — SEE IF DE IS ZERO
  179    B3           OR E          ;IF SO,LINE WASN'T FOUND &
  235    EB           EX DE,HL      ;HL=0 ELSE KEEP LOOKING
   32    20           JR NZ,TOP
  228    E4
   24    18           JR RETURN
  246    F6
```



VARLIST

by Charles C. Edwards

Here's another program to list those elusive variables. Instead of looking in the variable table at the end of your BASIC program, this utility scans through the actual program, thus catching ALL variables, not just those placed in the variable table by a RUN.

VARLIST will produce a sorted listing of all the variables used in a program. It will also tell you the type of each variable (integer variable, string array, etc).

Varlist can be used with any basic program having line numbers less than 60000. Rather than re-entering the program each time you want to use it, use the following technique (from the July 1979 TRS-80 Newsletter) to merge it with another program:

1) CLOAD the program to be analyzed
2) PRINT PEEK (16548), PEEK (16549) and write down these values
3) IF PEEK (16633) is 2 or greater then POKE16548,PEEK(16633)-2:POKE16549,PEEK (16634)ELSE POKE16548,PEEK(16633)+254:POKE16549, PEEK(16634)-1
4) CLOAD VARLIST
5) POKE 16548 and 16549 back to their original values

Both programs should now be in memory. To run VARLIST, ENTER "RUN 60000" and then go have dinner, mow the lawn, or do some other time-consuming task. I tested VARLIST by running it against Star Trek III.3 and it took approximatly 25 minutes to finish. It may be slow, but it beats doing it by hand!

```
 60000 DEFINTA-Z:DIMS$(199),A(99),T(25):FORI=0TO25:T(I)=3:NEXT:FO
 RI=0TO99:A(I)=0:NEXT:A=PEEK(16548)+256*PEEK(16549):CLS:PRINT@0,C
HR$(23):I=0
 60010 IFPEEK(A+2)+256*PEEK(A+3)=60000THEN64000ELSEPRINT@512,"NOW
  PROCESSING LINE";PEEK(A+2)+256*PEEK(A+3):A=A+4
 60020 P=PEEK(A):IFP=147THENGOSUB60050ELSEIFP>151ANDP<156THENGOSU
 B61000ELSEIFP=136THENGOSUB61100ELSEIFP=34THENGOSUB61200ELSEIFP=1
 90THENGOSUB61300ELSEIFP>47ANDP<58THENGOSUB61400ELSEIFP=0THENA=A+
 1:GOTO60010ELSEIFP>64ANDP<91THENGOSUB61500ELSEA=A+1
60030 GOTO60020
 60050 A=A+1:P=PEEK(A):IFP=0THENRETURNELSE60050
 61000 T=P-151:OL=0:C=0
 61002 A=A+1:P=PEEK(A):IFP=0ORP=58THENRETURNELSEIFP=44THENC=0ELSE
 IFP=206THENC=-1ELSEIFP>64ANDP<91THEN61004ELSE61002
61003 GOTO61002
```

```
 61004 P=P-65:T(P)=T:IFCTHENFORJ=OLTOP:T(J)=T:NEXT
 61006 C=0:OL=P:GOTO61002
 61100 A=A+1:P=PEEK(A):IFP=0ORP=58THENRETURNELSEIFP=34THENGOSUB61
200:IFP=34THEN61100ELSERETURNELSE61100
 61200 A=A+1:P=PEEK(A):IFP=0THENRETURNELSEIFP=34THENA=A+1:RETURNE
LSE61200
 61300 POKEVARPTR(A(0))+I,64:A=A+1:RETURN
 61400 A=A+1:P=PEEK(A):IF(P>47ANDP<58)ORP=68ORP=69THEN61400ELSERE
TURN
 61500 P=P-65:POKEVARPTR(A(0))+I,PEEK(VARPTR(A(0))+I)ORT(P):POKEV
ARPTR(S$(I))+1,PEEK(VARPTR(A)):POKEVARPTR(S$(I))+2,PEEK(VARPTR(A
)+1):L=1
 61510 A=A+1:P=PEEK(A):IF(P>64ANDP<91)OR(P>47ANDP<58)THENL=L+1:GO
TO61510ELSEIFP=33ORP=35ORP=36ORP=37THEN61520ELSE61530
 61520 L=L+1:IFP=36THENT=1ELSEIFP=37THENT=2ELSEIFP=33THENT=3ELSET
=4
 61525 POKEVARPTR(A(0))+I,(PEEK(VARPTR(A(0))+I)AND64)+T:A=A+1
 61530 P=PEEK(A):IFP=32THENA=A+1:GOTO61530ELSEIFP=40THENIFPEEK(VA
RPTR(A(0))+I)<64THENPOKEVARPTR(A(0))+I,PEEK(VARPTR(A(0))+I)+32
 61540 POKEVARPTR(S$(I)),L:IFI=0THENI=I+1:RETURNELSEFORJ=0TOI-1:I
FS$(J)=S$(I)ANDPEEK(VARPTR(A(0))+I)=PEEK(VARPTR(A(0))+J)THENPOKE
VARPTR(A(0))+I,0:RETURNELSENEXT:I=I+1:RETURN
 64000 GOSUB65000:K=0:CLS:FORJ=0TOI-1:IFK=15THENINPUT"HIT =ENTER=
  TO CONTINUE";Z$:K=0:J=J-1:NEXTELSEK=K+1:PRINTS$(J);:X=PEEK(VARP
TR(A(0))+J):Y=XAND7:IFY=1THENPRINT" STRING ";ELSEIFY=2THENPRINT"
  INTEGER ";ELSEIFY=3THENPRINT" SINGLE ";ELSEPRINT" DOUBLE ";
 64010 IFX>64THENPRINT"FUNCTION"ELSEIFX>32THENPRINT"ARRAY"ELSEPRI
NT"VARIABLE"
64020 NEXT:END
 65000 C=0:FORJ=0TOI-2:IFPEEK(15872)=83THENPRINT@512,"
               "ELSEPRINT@512,"SORTING..."
 65010 IFS$(J)>S$(J+1)THENGOSUB65100ELSEIFS$(J)=S$(J+1)THENIFPEEK
(VARPTR(A(0))+J)>PEEK(VARPTR(A(0))+J+1)THENGOSUB65100
 65020 NEXTJ:IFCTHEN65000ELSERETURN
 65100 FORK=0TO2:Q=PEEK(VARPTR(S$(J))+K):POKEVARPTR(S$(J))+K,PEEK
(VARPTR(S$(J+1))+K):POKEVARPTR(S$(J+1))+K,Q:NEXT:Q=PEEK(VARPTR(A
(0))+J):POKEVARPTR(A(0))+J,PEEK(VARPTR(A(0))+J+1):POKEVARPTR(A(0
))+J+1,Q:C=-1:RETURN
```

THE WATCH THAT REPLACES EVERYTHING.

In fact it even replaces the age-old myth that a timepiece loaded with talent has to be expensive.

The Casio C-80 has a list price of $50. It has been advertised nationally for $75. HARDSIDE's special price is only $44.95 (we even pay the UPS charges).

A CALCULATOR MADE FOR REAL FINGERS

Here's the first calculator chronograph that can be operated easily with your fingertips. No need to use a stylus or pen; the keys are far enough apart that even the broadcast fingers can work them.

Now you can add, subtract, multiply, divide - even perform chain calculations up to eight places, plus decimal. Numbers appear in crisp liquid crystal digits.

STOPWATCH, 2 TIME ZONES-AND MORE

The push of a button turns your calculator into a very talented stopwatch. You get 1/100 of a second accuracy, lap times, and up to 59.59 minutes of track-style timekeeping.

You also get two time zones; the second zone can be set for 12 or 24 hour time. Hours, minutes and seconds are displayed with reliable 15 second-a-month quartz precision.

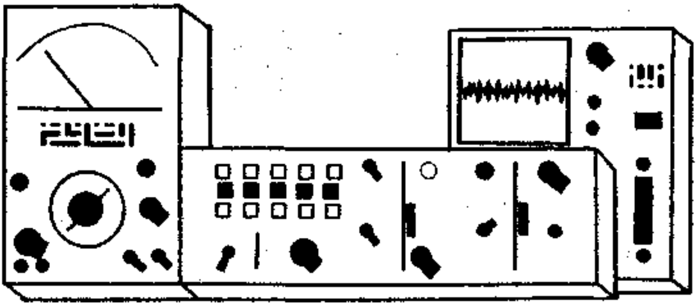
Press another button and the day of the week and date appear. And another for built-in night light.

FEATHER-LIGHT, YET RUGGED

The 1-3/4 oz. case and band are constructed of high impact black plastic. And so lightweight and comfortable, you may not realize that you are wearing it.

This Casio chronograph is built incredibly rugged to provide years of reliable service. A tough mineral glass crystal resists scratching. And the tightly fitted case and crystal provide water resistance for everyday use.

#99-01 .... $44.95



BUG REPORT

In your June 1980 issue, refer to the article, "Using Your Electric Pencil With the Radio Shack Lower Case Modification," on page 83, change the second and third modification addresses from: FO1BC FO1D9 to: FO1OBC FO1OD9

MICROSOFT

Create, debug, document & modify - 5 WAYS

UTILITY

FROM NEPENTHE

The UTILITY program consists of five modules to assist you in the creation and debugging of programs, the preparation of program documentation and the modification of program files. With UTILITY, you can: quickly locate any string of characters in your program, change any strings within your program, produce a line number cross-reference on screen or printer, select from three degrees of program compression, create a formatted, "structured" line listing.

UTILITY is available on disk for $29.95

Call 1-800-258-1790 (in N.H. 673-5144)

The Software Exchange

Add a whole new dimension to your computer!

FROM CAP ELECTRONICS

soundware™

Games and programs come alive with laser sounds, bounces, clicks, sirens, bird calls, music notes, tunes and whatever else your imagination dreams up. Supplied with three demonstration programs on tape, attractive amplifier with cable for direct hookup to your cassette recorder (no modification required) and complete instructions for creating your own sounds and tunes (including a 5-octave organ).

Let SOUNDWARE bring your TRS-80 to life for $24.95, Level II, 16K

(Requires 2 AA batteries)

The Software Exchange

6 SOUTH ST., MILFORD, N.H. 03055 (603) 673-5144

TRS-80 FORTRAN SUBROUTINES

by David E. White

I recently purchased the Microsoft Fortran package for the TRS-80 computer and have started to write a few programs in that language. The major advantages of Fortran compared to TRS-80 BASIC are 1) the greater speed of execution because it is a compiled language and 2) the ability to create truly independent subroutines and functions which can be kept in a library file and used as required in other programs. Some of the disadvantages are 1) the cumbersome three-step process for creating and running a program (i.e. editing, compiling, linking and loading), and 2) the lack of some of the useful capabilities of BASIC.

However, some of Fortran's disadvantages can be reduced. For instance, I find the Microsoft Editor to be rather a nuisance because it is a somewhat primitive line oriented editor and because it spends so much time in transferring text to and from disk during editing sessions. While this editor has the advantage of being able to edit files larger than can be stored in memory, very few Fortran files will ever be that large, especially if proper use is made of program subroutines. I have found that the Electric Pencil serves very well as a Fortran editor with much greater editing capabilities. The Fortran Compiler functions quite well without the line numbers as generated by the Microsoft Editor and even inserts its own sequential line numbers in the listing files (/LST). The only complication is the file extension which is '/PCL', as produced by the Electric Pencil, and needs to be '/FOR' for the Fortran compiler. The simple solution is to use the DOS 'RENAME' command to change the extension. A slightly more elegant approach is to go inside the Electric Pencil command file and change the string '/PCL', wherever it appears, to '/FOR'.

Also missing in Fortran are some of the very useful BASIC commands. For example: the random function, the clear screen command, the PRINT @ command, the graphic commands, etc.. Fortunately, most of these can be simulated using called Fortran subroutines and functions. A variety of functions are possible particularily by using the extended Fortran commands PEEK and POKE. Also because the Fortran programs are compiled, their speed is often greater than the equivalent BASIC version. I have found the following routines useful for my Fortran programming and hope that you might find them of value as well:

NAME	FUNCTION
1. CLEAR	Homes cursor and clears screen.
2. CURSOR	Moves cursor to specified screen position.
3. POINT	Sets, resets, checks graphic pips.
4. RND	Random number function.
5. PADDLE	Simulates a paddle using the arrow keys.

The first program is a simple example of using control codes and the A-type output conversion option to control an output device. In this case, two 'BYTE'-type variables are assigned control values and are then output to device 5 (the video screen) using the 'A' format option which transmits their literal value to the screen driver routines. The first byte (28) positions the cursor at location (0), and the second byte (31) clears the screen to the end of the frame. It is necessary that these bytes be output separately, using '2A1' rather than 'A2', to effect the proper control. Similiar procedures can be used to move the cursor, change to 32 character display mode and to exercise printer control.

```
C  ***  CLEARS SCREEN AND HOMES CURSOR  ***
      SUBROUTINE CLEAR
      BYTE BYTE1,BYTE2
      DATA BYTE1,BYTE2/28,31/
      WRITE(5,100) BYTE1,BYTE2
 100  FORMAT(1X,2A1)
      RETURN
      END
```

This next program is similar to the 'PRINT@' command in BASIC. However, instead of having to calculate the exact location number, one simply specifies the line number (from 1 to 16) and the print position (1 to 64). The subroutine then moves the cursor to that locaion for a subsequent read or write operation.

```
C  ***  CURSOR SETTING PROGRAM  ***
       SUBROUTINE CURSOR(LINE,POS)
       INTEGER LINE,POS,LOC
       BYTE LBYT,HBYT
C     ERROR CHECKING
       IF (LINE.LT.1) LINE=1
       IF (LINE.GT.16) LINE=16
       IF (POS.LT.1) POS=1
       IF (POS.GT.64) POS=64
C     DETERMINE NEW LOCATION AND BYTE VALUES
       LOC=15360+(POS-1)+64*(LINE-1)
       HBYT=LOC/256
       LBYT=LOC-256*HBYT
       CALL POKE(16416,LBYT)
       CALL POKE(16417,HBYT)
       RETURN
       END
```

This routine is equivalent to the 'SET', 'RESET' and 'POINT' commands in BASIC. The routine requires three arguments: the x and y locations as in BASIC (x:0-127, y:0-47), and a control code which selects the function performed. The control codes are as follows: -1, resets point; 1, sets point; 0, returns a value based on whether the point is currently set or not. Logical operations are used to check and change the bit which controls the graphic pips.

```
C  ***  GRAPHIC 'POINT' SUBROUTINE  ***
      SUBROUTINE POINT(X,Y,ICODE)
      INTEGER X,Y,ICODE,XBLK,YBLK,LOC,IJUMP,IBIT
      LOGICAL BYTE,BIT,PIP(6)
      DATA PIP/1,2,4,8,16,32/
C     VALUES: X 0-127;   Y 0-47;   ICODE -1,0,1
      IF (X.LT.0) X=127
      IF (X.GT.127) X=0
      IF (Y.LT.0) Y=47
      IF (Y.GT.47) Y=0
      IF ((ICODE.LT.-1).OR.(ICODE.GT.1)) ICODE=0
C     DETERMINING GRAPHIC BIT LOCATION AND VALUE
      XBLK=X/2
      YBLK=Y/3
      LOC=15360+64*YBLK+XBLK
      BYTE=PEEK(LOC)
      IF (BYTE.LT.1:8) BYTE=-128
      IBIT=X-2*XBLK+2*(Y-3*YBLK)+1
      BIT=PIP(IBIT)
C     MANIPULATING THE CODES
      IJUMP=ICODE+2
      GO TO (200,300,400), IJUMP
C     RESETTING PIP
  200 IF (BYTE.AND.BIT) BYTE=BYTE.XOR.BIT
      GO TO 500
C     CHECKING PIP
  300 ICODE=-1
      IF (BYTE.AND.BIT) ICODE=1
      GO TO 600
C     SETTING PIP
  400 BYTE=BYTE.OR.BIT
C     CHANGING MEMORY VALUE
      500 CALL POKM(LOC,BYTE)
      600 RETURN
      END
```

This subroutine generates pseudorandom numbers using a congruential method. The function generates real numbers uniformly in the range from 0 to X. Integer numbers in the range 1 through X can be generated by using INT(RND(X))+1. The procedure is based on a seed number whose value is protected by being defined as a member of a common block. The random numbers will follow different sequences depending on what happens to be in the seed memory location when the program is loaded. Conversely, one could initially specify the seed value through another program, which would then give a reproducible sequence of random numbers.

```
C  ***  RANDOM FUNCTION AS PER BASIC  ***
        FUNCTION RND(X)
        REAL A,BM,SEED
        DOUBLE PRECISION C,DM
        COMMON /XXXX/SEED
        DATA A,C,BM,DM/ 24298., 99991., 199017., 199017./
        SEED=DMOD(A*SEED+K,DM)
        RND=(SEED/BM)*X
        RETURN
        END
```

This routine converts the arrow keys into the equivalent of a paddle or joystick control. Depending on which arrow key or combination of arrow keys are currently pressed, the program returns the x and y direction indicated. This is accomplished by PEEKing at the appropriate keyboard memory location and checking the bits that are associated with each pressed key. This enables each key to be checked independently. For example, if both the left and the up arrow are pressed, then the program would return -1 for the x-direction and +1 for the y-direction. If no key, or contradictory keys are pressed, a zero value is returned. Similiar procedures could be used to read any key(s) on the keyboard.

```
C  ***  PADDLE  ***
        SUBROUTINE PADDLE(IDX,IDY)
C      READS THE ARROW KEYS FROM THE KEYBOARD
       LOGICAL KB,LUP,LDN,LLF,LRT
       DATA LUP,LDN,LLF,LRT/8,16,32,64/
       IDX=0
       IDY=0
C      PEEK AT KEYBOARD VALUES
       KB=PEEK(14440)
       IF (KB.AND.LUP) IDY=1
IF (KB.AND.LDN) IDY=IDY-1
       IF (KB.AND.LLF) IDX=-1
       IF (KB.AND.LRT) IDX=IDX+1
       RETURN
       END
```

FASTGAMMON

Backgammon is a two-person game.
In FASTGAMMON the computer is one player
(its men are O's) and you are the other (your men are X's).

Whether you are a beginner or an experienced backgammon player, FASTGAMMON can help you improve your game. First of all, you now have an opponent that is always ready and willing to play. There is no substitute for experience — the more games you play the more you learn about backgammon. But there is another unique method by which FASTGAMMON can help you improve your game.

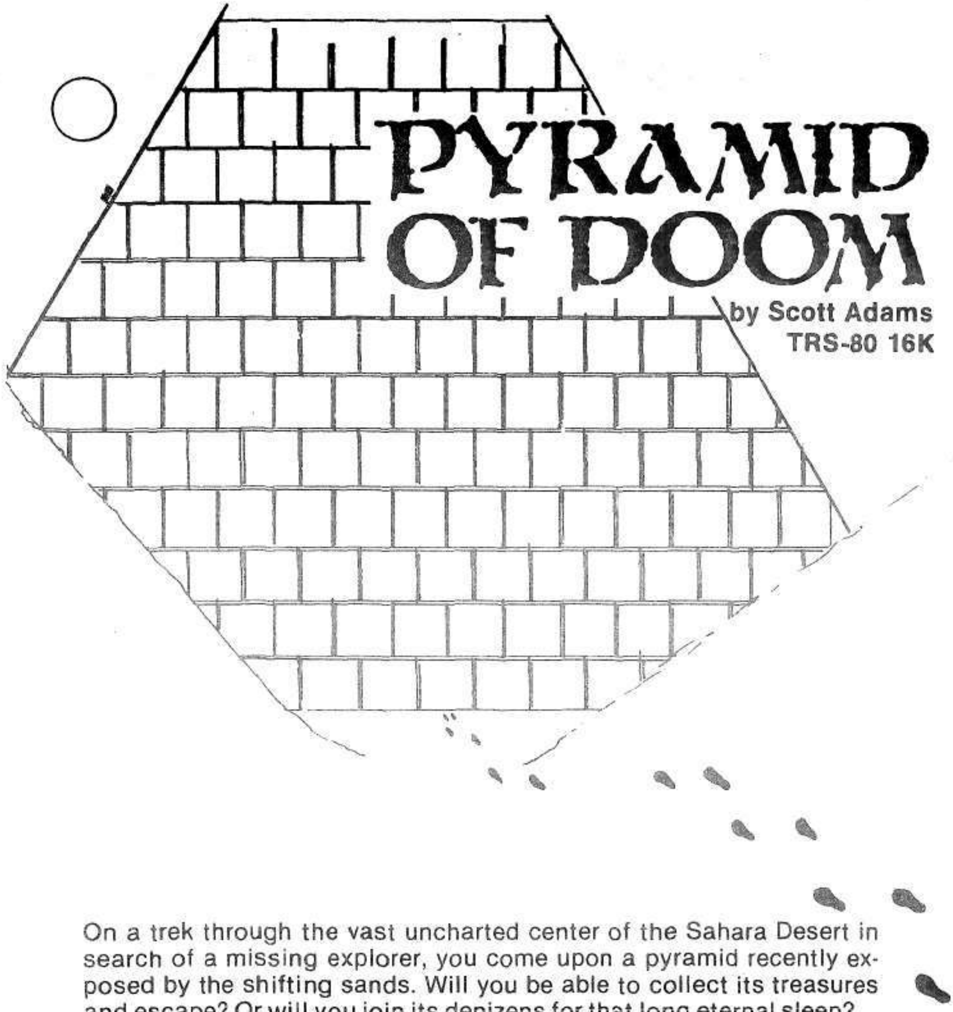
By selecting "2" when the starting message is displayed, the same sequence of rolls that occurred in the game just played (the last game for which you selected "1") will occur again.

HOW GOOD IS FASTGAMMON?

We make no claims that FASTGAMMON plays at an expert level. In fact, good players will notice obvious mistakes that the computer makes from time to time. The game of backgammon has many subtleties, and the algorithm programmed into version 1 of FASTGAMMON is a fairly simple one. There is an advantage that the computer has, however, over human players — the computer never forgets what it "knows". Due to oversights, distractions, or haste, human players often forget what they know. This fact is the edge that makes FASTGAMMON competitive against good players. Level II 16K $19.95

QS *Quality Software*

TSE
The Software Exchange
6 South Street, Box 68, Milford, NH 03055 603-673-5144

PYRAMID
OF DOOM
by Scott Adams
TRS-80 16K

FOR TSE Order Form SOFTWARE ONLY

Accts Rec II Disk 32K	69.95
Adv Pers Fin Disk	24.95
Adventure Sample = 0	5.95
2 Adventures on Disk	24.95
☐ Land & Pirate	
☐ Mission & Voodoo	
☐ Count & Odyssey	
☐ Fun House & Pyramid	
3 Adventures Disk	39.95
☐ Land Pirate Mission	
☐ Count Voodoo Odyssey	
Adventures on tape	14.95
☐ Count	
☐ Fun House	
☐ Land	
☐ Mission Impossible	
☐ Pirate's Cove	
☐ Pyramid of Doom	
☐ Strange Odyssey	
☐ Voodoo	
Adventure (Microsoft)	29.95
Airmail Pilot	7.95
Air Raid	9.95
Alien Invasion (sound)	cass. 9.95
Alien Invasion (sound)	disk 14.95
Amateur Astronomy Handbook	14.95
Amateur Radio Disk	24.95
Amazin' Mazes	9.95
Android Nim (sound)	14.95
APL	disk 34.95
APL	w/book 49.95 + $3
APL	tape 14.95
APL	book only 15.50 + $3
Appointment Log	9.95
Automated Disk Directory	14.95
Barricade	9.95
Basic Handbook	14.95 + $1
Basic Statistics	9.95
Basic Styles Handbook	5.95 + $1
Bee Wary (sound)	14.95
Binders	4.95 + $1
Biorhythms	4.95
Bismarck	49.95
Bridge Challenger	14.95
Cards of Fortune	7.95
Cassettes	
☐ C-10	10 for 6.95 + $1
☐ C-20	10 for 7.95 + $1
Casino Anthology	7.95
Challenge (sound)	9.95
Chess Companion	7.95
COMPROC Command Processor	tape for disk only $19.95
Cribbage	7.95
Data Management System (CCA)	74.95 + $2
Datestones of Ryn	
Tape	14.95
Disk	19.95
Diskettes	
BASF	☐ 5 for 24.95 + $1
	☐ 10 for 39.95 + $1
	☐ 20 for 69.95 + $2
	☐ 100 for 299.00 + $3
Dysan	box of 5 29.95 + $1
Dynamic Data Base	39.95

Diskette Storage Box	5.00 + $1
Dosort 32K or 48	tape 34.95
DSM Sort Utility for disk	75.00
Editor/Assembler Plus 16K	tape 29.95
Electric Pencil	tape 100.00
Electric Pencil	disk 150.00
Educator Assistant	9.95
8080-Z80 Conversion	14.95
Electronics Assistant	9.95
End Zone II	9.95
Fastgammon	19.95
File Manager 80	49.95
Floppy Armour	box of 5 4.95 + $1
Floppy Disk Diagnostic	24.95
Fortran/Assembler	150.00 + $5
Fortran	80.00 + $2.50
Assembler	80.00 + $2.50
Galactic Empire	14.95
Galactic Empire/Trader	disk 29.95
Galactic Revolution	14.95
Galactic Trader	14.95
Galactic Trilogy	disk 39.95
GSF 16K,32K,48K	24.95
Ham Radio	9.95
Histograph/Scattergram	9.95
Home Financial Management	9.95
I Ching Level II 16K	7.95
Infinite BASIC by Racet	49.95
Business add-on	29.95
Intro TRS-80 Graphics	7.95 + $1
Invasion of Orion	
Tape	19.95
Disk	24.95
Inventory II.3 Disk	79.95
Inventory S	tape 24.95
Inventory S disk	w/invoicing 59.95
	w/o/invoicing 39.95
Investment Portfolio	49.95
IRV	
Tape	24.95
Disk	29.95
Journey Center Earth	7.95
Kamikaze	7.95
Keyboard	9.95
Kriegspeil	7.95
KVP	14.95
KVP on Disk	19.95
KVP 232	24.95
Learning Level II	15.95 + $1
Level I in Level II	14.95
Level III BASIC	49.95
Life Two (sound)	14.95
Loan Amortization	19.95
Lost Dutchman's Gold	9.95
Lower System	9.95
Machine Lang. Mon. (RSV.2)	26.95
Machine Lang. Mon. (RSV.2D)	29.95
Magic Paper Calculator	14.95
Mail List II Disk	99.95
Marlock's Tower	
Tape	14.95
Disk	19.95
Mastermind	7.95
Mean Checkers Machine	tape $19.95
Mean Checkers Machine	disk 24.95

Microchess 1.5	19.95
Microtext Editor	9.95
Moving Signboard	9.95
Mu-Math	74.95 + $3
NEWDOS	49.95
NEWDOS +	99.95
NEWDOS 80	149.00
9 Games/Preschool Child	9.95
Numerology 32K	disk 14 95
On-Line Invoicing	disk 39.95
Pathways Through the ROM	19.95 + $1.
Payroll Disk 32K	39.95
Pencil Pal	35.00
Pentominoes (sound)	7.95
Periodical Cross Reference	tape 14 95
Periodical Cross Reference	disk 19.95
Personal Finance	9.95
Pigskin	9.95
Pork Barrel	9.95
PR Dogfight	7.95
Preflight	20.00
Print Spooler	disk 24.95
Remodel & Proload 16K,32K,48K	34.95
Renumber	7.95
Renumx	24.95
Rescue at Rigel	
Tape	19.95
Disk	24.95
Roots	disk 19.95
RPN Calculator	9.95
RSM-2	26.95
RSM-2D	29.95
RX (disk)	24.95
Santa Paravia & Fulmaccio	7.95
Sargon Chess	19.95
Sargon II	29.95
Sargon Handbook	15.95 + $1
Scripsit	
Tape/Disk	95.00 + $2
Secrets of the Tarot	9.95
Small Business Bookkeeping	tape 24.95
Small Business Bookkeeping	disk 29.95
Small Business Bookkeeping	journal 7.00
Snake Eggs (sound)	14.95
Space Battles	tape 14.95
Space Battles	disk 19.95
STAD	24.95
Star Fleet Orion	
Tape	19.95
Disk	24.95
Star Trek III.4	14.95
ST80 Smart Terminal	49.95
ST80D Smarter Disk	79.95
ST80 III	150.00
ST80UC	24.95
Super Simon	$9.95
System Copy	9.95
T Short	9.95
Taipan	9.95
Tape Recorder Alignment Kit	9.95

Temple of Apshai	
Tape	24.95
Disk	29.95
Text 80 Disk	59.95
Time Series Analysis	14.95
Time Trek	14.95
Ting Tong Level II 16K	9.95
Tiny Comp	tape 19.95
Tiny Comp	disk 24.95
Troll's Gold	4.95
TRS-80 Assy. Lan. Pro.	3.95 + $1
TRS-80 Disk & Other Mysteries	22.95 + $1
TRS-80 Interfacing	8.95 + $1

TRS-80 Opera Theatre (sound)	9.95
with challenge on disk	19.95
Tycoon	7.95
Typing Tutor	19.95
Warfare	7.95
Wordo	14.95
X-Wing Fighter II	9.95
XREF	19.95
Y-Y Bar by Racet	14.95
Z-80 Chip Poster	3.99 + $1
Z-80 Gourmet Cookbook	14.95 + $1
Z-80 Handbook	4.95 + $1
Z-80 and 8080 Assembly	7.95 + $1

The Software Exchange

6 SOUTH STREET MILFORD, NH 03055 Order Number: 603-673-5144

Level II software available on disk for a $5.00 (per order) medium charge. This extra fee is for any number of programs transferred to disk from tape when you order. If the order exceeds the capacity of a single disk, we absorb the extra cost.

Please state level and memory size on order form . . . otherwise, we automatically ship Level II cassettes.

Be sure to include handling charge and any additional charges when figuring your total. All orders *shipped with 24 - 72 hours. Prices subject to change without notice.*

☐ UPS ☐ 1st Class ☐ Charge ☐ COD

Charge card account number

Signature . . . .

Expt. Date . . . . **Inter.#** . . . .

Charge customers: please fill in account information above and below

Name . . . .

Address . . . .

City . . . . **State** . . . . **Zip** . . . .

ALL SOFTWARE SOLD ON AN AS-IS BASIS WITHOUT WARRANTY. TSE assumes no liability for loss or damage caused or alleged to be caused directly or indirectly by equipment or products sold or exchanged by them or their distributors, including but not limited to any interruption in service, loss of business or anticipatory profits or consequential damages resulting from use or operation of such equipment or software. We are not responsible for typographical errors anywhere in this publication

DO NOT FORGET SHIPPING CHARGES

Shipping Charge - Add $1 Disk Charge - Add $5

All C.O.D. and overseas orders must add $5.

IMPORTANT ANNOUNCEMENT

Beginning with the next issue, PROG/80 will be combined with **SoftSide** in a full-size format. So, in addition to the advanced programming tips and techniques you've come to expect, you'll also be getting:

Feature programs for the TRS-80, APPLE and ATARI. Special departments on...

- ***Graphics**
- ***Music & Sound Effects**
- ***Programming Hints**
- ***Take-A-Parts**
- ***And Much More!**

Plus, it's monthly!

PROG/80
A SoftSide Publication
P.O. Box 68, Milford, NH 03055