THE SNUGLETTER

FROM THE SOUTHERN NEVADA USERS' GROUP

VOLUME 2, NUMBER 4 SEPTEMBER/OCTOBER 1984

******************** LETTER FROM THE EDITOR ***********************

It has been a considerable time since many of you have heard from SNUG if you have not attended a recent meeting. With the resignation of many original officers the group found itself in a difficult position. There was no one to assume responsibility for the operation of the necessary and vital functions to keep the group going. But a core of faithful original members and quite a number of those hoping-to-join kept the group interest up. New officers were elected in August and the necessary records, i.e., membership roster, etc., were located allowing the reestablishment of newsletter mailing to all the original members.

This also allows a reminder to all those members who are up for renewal of their membership. The annual dues are still $18 and will allow the participation in any of the group's activities.

There are quite a number of items of interest to those who haven't been to a meeting in some months. We now have expanded our library to over 450 programs through the aid of the Washington, D.C. group. We also have TI Forth in the library as a public domain release by Texas Instruments. We also should have enhancement programs for both TI-WRITER and MULTIPLAN, as well as, an assembly language debugger - all from TI in care packages to the users' groups. The newsletters from other users' groups are available for reading at the meetings with copies of the best articles available on a first served basis for check out. Part of the meeting time is turned over to use by the members in special interest sessions, such as, beginners BASIC instruction, Forth Interest Group, etc. Expectations and plans are for these groups to be expanded in number and scope as members express interest in other areas.

"NEXT MEETING NOVEMBER 12, 1984"

MONDAY - VERTERANS DAY AT 6:30 PM

CHARLESTON PLAZA LIBRARY

FORTH...an exciting alternative
by John Martin

ARE you getting tired of BASIC? Do you get frustrated by the often lengthy time delays while doing calculations or searching files? Is assembly language too alien for you to conquer? The answer to these and many other problems may just be FORTH. Programming in FORTH offers much of the speed of assembly, but is almost as friendly to use as BASIC. In this column and future columns, I hope to show that you can *program in FORTH* almost as easily as in BASIC.

There are certain requirements to your system to program in FORTH.

1. EDITOR/ASSEMBLER CARTRIGE
2. 32K MEMORY EXPANSION
3. DISK DRIVE SYSTEM
4. the FORTH system disk
5. a FORTH "work" disk

It is also helpful, but not necessary, to have a printer.

Here is the proceedure to "boot" FORTH.

1. Insert EDITOR/ASSEMBLER cartridge
2. Insert the FORTH system disk in the disk drive
3. Turn on the EXPANSION SYSTEM and the computer
4. Select EDITOR/ASSEMBLER from main menu
5. Select LOAD and RUN option from E/A menu
6. When prompted for FILE NAME, type DSK1.FORTH

As soon as the disk drive stops running, you should have the FORTH menu screen on your monitor. The selections you make from this screen will depend on the type of programming you want to do. In any case, you will need to at least load one of the editors. TI FORTH has two editors. You must decide which one you want to use since only one can be in memory at a time. Unless you have a good monitor and you don't mind straining your eyes to read the tiny letters used by the 64 column editor, I suggest using the 40 column editor. To load an editor, or any of the other options, you simply type it's name. The 40 column editor, for instance, is loaded by typing -EDITOR. Be sure to include the "minus" sign in the name.

As soon as you have loaded all the necessary options for your application, be sure to remove the system disk and insert your "work" disk. It would be a good idea to make a backup of your system disk just in case you forget this step sometime.

The first step to programming in FORTH should be to prepair a "work" disk If you only have one disk drive, the proceedure is as follows.

1. use the DISK MANAGER MODULE to initialize a disk
2. copy FORTH and FORTHSAVE onto the disk (do not copy SYS/SCRNS)
3. boot the FORTH SYSTEM DISK and copy screens 0 through 5 to the work disk

Step 3 will make it possible to make backup copies of your work disk with the DISK MANAGER MODULE and place the error messages on the work disk.

The procedure for copying screens 0 through 5 to your work disk with a single disk drive is as follows:

1. boot your FORTH system disk
2. Type 0 DISK_LO !
3. Type 0 BLOCK UPDATE 1 BLOCK UPDATE 3 BLOCK UPDATE 4 BLOCK UPDATE 5 BLOCK UPDATE
4. Place your work disk back into the drive and type FLUSH ABORT

Another way to do the same thing would be to define a word to carry out these same functions for us. Such a definition might look like this.

```
SCR #6
 0 ( SET UP WORK DISK )
 1
 2 : SET BLOCK UPDATE ;
 3
 4 : MSG CR ." AND PRESS ANY KEY " KEY DROP ;
 5
 6 : SETDISK CLS 0 0 GOTOXY ." PLACE SYSTEM DISK IN
 7DRIVE " MSG
 8 0 DISK_LO ! 0 SET 1 SET 6 3 DO I SET LOOP
 9 CLS 0 0 GOTOXY ." REMOVE SYSTEM DISK FROM DRIVE "
10 CR ." PLACE WORK DISK IN DRIVE "
11 MSG FLUSH ABORT ;
12
13
14
15
```

On this screen we have three definitions. The first word, SET, is defined as BLOCK UPDATE. BLOCK expects to have a screen number on the stack when it is executed. It then calls up that screen from a disk and leaves the address where it was stored in memory on the stack. The word UPDATE marks the most recently accessed block as having been updated. This means that it can be flushed to the disk.

The next word, MSG executes a carriage return (CR) and prints the text following the ." to the screen. KEY is similar to the basic word ACCEPT in that it halts the program execution until there is a keystroke. This keystroke is then left on the stack. DROP drops the last number from the stack

GOTO PG.5

******************** ON ENTERING PROGRAMS ******************

When I am entering programs from a magazine, such as "Home Computer Magazine" there are several things I have found be helpful. First I use my Extended Basic Module whenever possible. It is faster during LIST of program lines. It also has that very helpful REDO function (keys FCTN and 8) to recall one of those long lines which had a typing error which caused the BASIC interpreter to return a SYTAX ERROR when the ENTER key was pressed. The EXT. BASIC module's REDO will bring that offending line back with all of the errors and allow correction of typos with the editing keys. (That saves retyping the line.)

A second feature I have found is the use of the NUM statement. (This works in both TI and EXTENDED BASICs.) NUM, of course, is invaluable in automatically generating line numbers during program entery. However, I have also found it to be useful for program editing. When NUM is used with an existing line number the line number plus whatever is on that line will be displayed with the cursor on the first positon on the line after the number. Also the line number is displayed WITHOUT the greater than '>' symbol before the line number signifying that there is currently information on the line. When NUM generates a new line number the '>' symbol precedes the number. When NUM is used to display a line all of the editing functions normally available can be used to change the line. When ENTER is pressed the line is filed away and the next line number in the NUM sequence is displayed, either another currently existing line or a new line number (with the '>' preceding a new number). Thus an entire file can be reviewed, using NUM, by needing to ess only enter instead of the FCTN and X or E (arrow) keys. I said that all of the editing functions worked. However one does not work in quite the same fashion. The FCTN and 4 (CLEAR function) when used will stop the line display/listing and does NOT erase a line.

I have found these techniques to be quite useful when when entering BASIC programs.

RUDY JOHNSON - Southern Nevada Users' Group

***** BLANK SPECIAL FUNCTION KEY STRIPS *****

Again from the Sun Coast Beeper comes some neat information regarding how to write on those blank (blankity-blank) special function key strips. Try using a Sanford (brand name) SHARPIE Extra Fine Point Marker. It works great on this type of finish, even on glass!!

***** BLANKETY BLINKING CURSOR *****

Again from the Atlanta 99/4A Computer Users Group Call Newsletter is a very short entry you can make to prevent the cursor from blinking. This won't work in Basic, but if 'i're in Extended Basic try this:

```
10 CALL COLOR(0,11,1)
```

***** PARALLEL INTERFACE ON THE RS-232 CARD *****

From the Washington DC Area TI Home Computer Users Group Newsletter comes the following article pertaining to the latest from TI on how to connect the PIO output to a printer:

Connect TI pin 1 to CEN connector pin 1, TI pins 2-9 to CEN pins 2-9, TI pin 10 to CEN pin 11, TI pin 11 to CEN pin 29, and TI pin 16 to CEN pin 16. This should work for many printers (including TI and Epson). When outputting to a Smith-Corona model TP-1, the signal on pin 1 has to be inverted with the use of an electronic circuit or you may buy a special cable with this inverter built-in. To use with an Okidata printer, two lines require the inverted signals. There is also a special cable for this printer. These cables are advertised in the HOME COMPUTER magazine (old 99'er magazine).

************************* USER'S REPORT *************************

Subtitled: User's Report of the CorComp Disk Controller and 32K Memory Expansion (And the Digital Equipment RX180 Dual Disk Drive Unit) by RUDY JOHNSON of SNUG

When I decided that the time had come to expand my TI I still was not convinced that TI's expansion system was the way I wanted to go. The disk drive and controller which they offered was dated in comparison to the rest of the computer industry in the disk performance. I continued watching and waiting for something better.

When I saw the announcement by CorComp of their upgraded disk controller I felt that this was the item that would give the TI the zing which I wanted. When I saw the surplus DEC RX180 Dual Disks for under $300 I jumped at the opportunity. I was really in a stew though since I really wanted the CorComp 9900 MicroExpansion system. But I became frustrated waiting for the package (the cassette load times were becoming unbearable.) When I heard that another member had an extra p-box for sale I started figuring - it would be about $100 more but $100 more going with the box over waiting for the MicroBox. But it it would give me further expansion capabilities not possible with the MicroBox. So off I went to Comptuter Magic to find that the only CorComp controller they had was faulty in the TOOL SHED Utility package. So more waiting. Finally CM called! They had the DD Controller and a 32K card. OK! "Be there Friday night or Saturday morning!" Friday evening came and I went over there. I hauled my p-box and disk drives along. We connected all the things in, fired it up and both drive select lights came on and stayed on! Well! Invert the plug on the controller card and try again. Throw the switch and everything looks good! Everything runs perfectly!

So what have I found in the 6 weeks that I've had it. First, one needs the 32K memory expansion to use CorComp's Disk Manager, which is supplied on disk and NOT in a module. But the Controller and Manager make a sweet package when teamed with the DEC drives! I can format a disk as either single or double density (that's either 90K or 180K) per drive (The controller will handle up to four drives in any combination including double sided.) Since I've had no problems with the hardware it would be easy to forget about it. However there is the previously mentioned TOOL SHED utility package. It includes routines for reading and writing to both cpu and video memory, writing values to video registers, moving blocks of memory and executing assembly language subprograms by address. These are some powerfull utilities and were only available in the MiniMemory or Editor/Assembly modules. The demo program which is included on the Manager disk gives some idea of the possibilities for games and routines tied to the video display. And the DEC drives have worked flawlessly. I've started to learn TI Forth and have had the drives dancing around quite a bit going from one part of the disk to another. With the CorComp controller there was no need to make the modifications which are needed when the TI controller is used.

The disk controller comes with a 90 page instruction manual so you can guess that I have not covered everything which is available in this unit. Most of the varialbes in disk control can be set in the configuration of the Manager - even the disk interlace (for those who are interested in such things). The defaults for the drive configuration for formatting and copying can be preset to whatever the system requires. However they can be reset for any particular requirement at any operation.

The only problem I've had with any of the hardware is the LED activity light on the 32K card is no longer working. The light was faint to begin with. But since the memory still works I can live without it (the power on light on my computer has been out for months and it doesn't bother me.)

In summary I can say that I am quite satisfied with the disk control-e we don't really care which key was struck.

SETDISK first clears the screen (CLS), Then positions the cursor at column 0 and row 0 (0 0 GOTOXY), prints the system disk message, and executes MSG. 0 DISK_LO ! zeros the variable DISK_LO which controls the lowest screen number that you can write to. The next few words read the information from screens 0, 1, 3, 4, and 5. this information is transfered from the disk and marked as updated by SET. Since there are only 5 buffers available on the system, I didn't include screen 2 because each screen uses 1 buffer. Screen 2 doesn't do anything.

The screen is then cleared again and the work disk prompts are displayed. Flush puts the information onto the work disk, and ABORT clears the stack of the addresses left by multiple calls of BLOCK. Clear as mud, right.

If you are going to use this method, first clear screen 6 by typing 6 CLEAR . Next, type 6 EDIT. Now type in the application as listed here. When you have finished, press FCTN 9 (BACK), and type 6 LOAD FLUSH.

The application is now compiled in memory. When you want to use it, simply type SETDISK and follow the prompts. I am assuming you have already copied FORTH and FORTHCOPY with the DISK MANAGER MODULE.

If you are fortunate enough to have two disk drives, your task is much easier. Remember that FORTH numbers everything from zero. This makes DSK1 = DR0 and DSK2 = DR1. Here is the procedure for formatting and setting up a work disk with two drives:

1. Place the system disk in DR0
2. Place a new unformatted disk in DR1
3. Type 0 DISK_LO ! 180 DISK_HI !
4. Type 1 FORMAT-DISK
5. Type 0 90 20 SMOVE

That's all there is to it. If you have already formatted your disk with the DISK MANAGER MODULE, you can omit step 4. If you have already put applications on the disk, PLEASE omit step 4. Remember to keep a write protect tape on your system disk at all times when using it to format work disks.

Now that you know how to get set up, you will need to know a bit about the programming proceadure itself. Here is a short program that should help to demonstrate how to write a program in FORTH. Before typing in this application, first load -EDITOR -VDPMODES and -GRAPH.

Now, carefully type the following:

```
: STARS GRAPHICS2 1000 0 DO 256 RND 192 RND DOT
        LOOP TEXT ;
```

Be sure to leave a space after the colon and before the semicolon.

When the computer says "ok", type STARS (the word you have just defined). If you typed it in correctly, you should have a pretty impressive demonstration of the speed of FORTH. This word causes the computer to put 1000 "stars" on the screen in random locations.

Let's run through this definition word by word to see how it works. The first "word" is the colon. this tells the computer to begin compiling a new definition into the dictionary. The next word is STARS which is the word we are defining. GRAPHICS2 initializes the bit map mode and turns the screen black. 1000 and 0 are the ending and begining values of the DO LOOP respectively. Notice that we put the numbers on the stack first and then the operator. DO is similar to the basic word FOR in that it begins a loop procedure. 256 RND generates a random number between 0 and 255. This is used for the dot column parameter. 192 RND generates the dot row value in the same way. Both of these random numbers are left on the stack for use by the word DOT which turns on the pixel located at those coordinates. LOOP tests the value of a variable named "I" which keeps track of the number of times the loop has executed. If that value is less than the upper limit of the loop, the words between DO and LOOP are executed again. LOOP has a similar function to the BASIC word NEXT. If the value of I is equal to or greater than the upper limit of the loop, the next word in the definition is executed. In this case, the next word is TEXT, which puts the computer back into text mode. From text mode you can define or execute more words. The last word in the definition is the semicolon. The semicolon acts as a delimiter for the colon. That is, it tells the computer to stop compiling this definition.

A similar program in BASIC could be written like this.

```
100 CALL CLEAR
110 CALL COLOR(2,16,1)
120 CALL SCREEN(2)
130 FOR STAR=1 TO 1000
140 CALL HCHAR(INT(24*RND)+1,INT(32*RND)+1,46)
150 NEXT STAR
160 END
```

As you can see, FORTH is much more compact. Also, the FORTH version is in BIT MAP MODE. This means that there are 256 by 192 pixels available to turn on or off individually, whereas in BASIC, you only have 8 X 8 grids of pixels to turn on or off in patterns that have to be predefined. This means that you have only a 24 by 32 grid of character positions available.

The next thing you might like to do is to save your application to the disk. To do this, you must first place your definitions on a FORTH "screen". There are several

GOTO PG. 6

steps involved in doing this. They are as follows:

1. Type the screen number you want to use and the word CLEAR
2. Type the screen number and the word EDIT
3. Type your definitions on the 16 lines provided on the screen
4. Press FCTN 9 to return to immediate mode
5. Type FLUSH
6. Type the screen number and the word LOAD

You may use any screen number between 20 and 89 without worrying about writing over any of the FORTH screens on your work disk.

Step 6 will compile the definitions you have entered on the screen into the dictionary. If you want to change a definition, just type the screen number and EDIT, make your changes, press FCTN 9, type the screen number again and the word LOAD. Once you get used to these steps, you will find that writing in FORTH is very much like writing in BASIC.

One more topic I would like to cover. There is a word in FORTH called BSAVE that will, along with it's counterpart, BLOAD, save you a lot of time when you first boot your FORTH work disk. BSAVE saves the compiled version of your application, along with all of the options you have loaded from the menu screen, to the disk in what is called binary image format. Since this format is much easier for the computer to read, your entire application will load in just slightly longer than it takes to get the menu screen normally.

The recommended syntax for BSAVE is ' TASK nn BSAVE .

The apostrophe (pronounced "tick") places the memory address of the word TASK on the stack. TASK is simply a word that marks the boundary between the FORTH kernal and the applicaton. By typing FORGET TASK, you can remove an entire application from memory and have only the kernal loaded. The nn represents a screen number that you select for the BSAVE to begin. Any valid screen number over 19 will be satisfactory. Keep in mind that the BSAVE will generally use 5 or more screens, so be careful that you don't accidently write over your source code. BSAVE also leaves the next available screen number on the stack. If you want to know what that screen is, simply type a period and enter.

The other side of the BSAVE coin is BLOAD. The syntax for BLOAD is nn BLOAD. Where nn is a screen number where you have previously BSAVED an application. One of the most effective ways to use BLOAD is to put it on screen 3 which is designated as the BOOT screen. Each time you load FORTH, screen 3 is loaded immediately after the FORTH kernal loads.

In order to place our BLOAD on screen 3, we first have to have at least an editor loaded from the FORTH SYSTEM DISK. Next, type 3 DUP CLEAR EDIT. Now, on any convenient line, type nn BLOAD. If you want to have your application boot and run automatically, you can also type the name of your application on a line. Next, press FCTN 9 and type FLUSH. The next time you boot your FORTH disk, your application will automatically be loaded also.

I hope this article has helped you to get started in FORTH. If you are interested in learning more, see me or one of the other members of the FORTH group at the next SNUG meeting. We usually set the time and place for the next FORTH meeting during the regular SNUG meeting. See you there!

S. N. U. G.
P. O. BOX 492Ø
LAS VEGAS, NV 89127

TO:

FIRST CLASS