BLUEGRASS AREA 99er's

BYTEMONGER

LEXINGTON, KENTUCKY - Heart of the Bluegrass - DECEMBER 1984

TOPIC: TI-TELE COMMUNICATIONS

Equipment for computer communications will be discussed and demonstrated at our meeting to be Held on THURSDAY, DECEMBER 6, 1984. The meeting will be held, as usual at the Kentucky Utilities Operations Center - 500 Stone Road beginning at 7:00 PM.

Included in the demonstration of the equipment will be accessing TEXNET and the SOURCE which provide many services including TI programs which may be downloaded. We will also be demonstrating local telephone communication, data downloading and uploading. We also hope to contact one of the Kentucky Bulletin Boards to demonstrate the services which are offered. We hope to have both the accoustic and the direct connect modems to demonstrate.

As always, the informal small group discussions and program swapping which follow the formal presentations are as beneficial and allow many an opportunity to get questions answered and problems solved.

EDITORS COMMENTS

After the first two issues of a much needed means of communication the BYTEMONGER format is beginning to solidify into something of which the organization may be proud. Beginning with this issue we feature a column titled HOLDING FORTH by John Schmidt. His introductory article to this continuing series titled PRESSING FORTH promises some helpful and informative future tutorials on the powerful language of TI-FORTH. Because we include both of these articles as well as our normal communications, the size of the BYTEMONGER has increased. This may be permanent or temporary depending upon the future contributions of others within the organization. We welcome John's contributions and anticipate that they will be helpful not only to our group but to many other TI clubs who also receive our publication.

The change in meeting night from Tuesday to Thursday will mean that the BYTEMONGER will be received by our members several days before the meeting. The first two issues for our meeting just happened to be published on the weekend before two Monday holidays for the postal service and many members received issues as late as the Tuesday mail. Editor.

MEETINGS CHANGED TO THURSDAY

Gerald Wells made a presentation at our last meeting which included a discussion of the membership survey recently completed. The program which he wrote to analyze the data of each member who responded was discussed and demonstrated. Printouts from that program were produced and available to all members which listed members by primary and secondary personal interest. Tabulations of members which had various hardware and peripherals were also produced and distributed, along with a listing of the data analysis program. The Data Program will be continuously updated for new members so that the information is available for decisions affecting all of the members. Any member who has significant changes as a result of upgrading their equipment may provide this information to the Group secretary so that their data file may remain current.

One significant piece of information derived from the survey was that a number of members had serious conflicts with the Tuesday night meeting, and the membership voted to change the meeting night to the FIRST THURSDAY of each MONTH. The data showed Thursday as the only night in which no member had a conflict.

In other business, the membership authorized the purchase of all of the HOME COMPUTER MAGAZINE programs on Cassette for the Library as well as a number of public domain diskettes which have recently become available.

BLUEGRASS AREA 99er USERS GROUP NEWSLETTER

The BYTEMONGER is a monthly publication. Correspondence should be addressed to Editor, P.O. Box 11866, Lexington, Kentucky 40578-1855. Advertising rates are included with suppliers copies. Reports and Technical contributions are solicited from members and others.

Articles published in the BYTEMONGER may be quoted if proper credits are given.

Editor.....Don MacClellan
Able assistance.....Jim Hughes & Gerald Wells

*** LOW LOW NOTES ***

Don MacClellan

In the October issue we printed a low note program written by TIGERCUB's Jim Peterson which allows the programmer to produce low notes two octaves below those which TI says is the limit of the 99/4A. The November issue printed a change in the CALL SOUND statement to correct a editing error. I became curious as to how low a frequency was possible and began playing with some modifications to the program which I think you will find interesting.

Mr. Peterson is using a characteristic of the noise generator, or 4th note, to generate the low tones. The notes in the scale are related by the equation F=F0(2^(1/12))^N. There are several good articles in 99er Magazine about musical notes, notably one by Norma and John Clulow in Vol.1 No.2. Only the two noises -4 and -8 are affected by the third note specified and Mr. Peterson is using this quality to generate a tone as the Noise (or 4th note) by selecting a value he calls a code for the third note. There must be some mathematical relationship which generates his value of F=1652 but I have not yet discovered it, however, it is a value which is very near 1661.22 the frequency of G#4. I spent one afternoon plugging equations in the program in an effort to print not only the appropriate CODE to generate the LOW NOTE but to print out its frequency. Many of you will readily see how to do it. I never did. In frustration I used the simple relationship between his SEED F=1652 and the frequency of the first note played - the "absolute lowest TI note" of 110 cycles per second. This you will see in line 230.

There is a practical limit to how low a frequency can be reproduced. The best and most expensive stereo speakers often can reproduce low notes in the 15 cps range and below that sound is inaudible to most. Notes in that low range are felt rather than heard by low frequency vibration or air pulsations. It can hardly be expected that a 2-4 inch television or monitor speaker could reproduce anything much below 50 cps and certainly if anything it would be very distorted. It should be noted then that any notes much below 50 cps which you hear and which are not reproduced on a 12-15 inch stereo speaker are distorted and will begin to sound the same or become inaudible. I therefore selected three octaves as a limit. Since we are cycleing through a 12 note octave and since we want to do it three times a simple basic loop is used to repeat the cycle three times and another loop within to step through the notes. The loop can be repeated a fourth time by changing line 170 which will illustrate the sound limits of your hearing and your monitor.

The program is kept in Basic so that anyone can use it but you will find that it runs much faster in XB. The balance of the changes are just for cosmetic effect; a few column labels, ASCII 37 replaces "Flat" and 39 is just for spacing, any other screen color is a good substitute for cyan. If you want a printout for easy reference to the CODE's for programing music or low notes, I have included a simple print statement for a parallel printer; just remove the REM statements, or modify for your printer.

```
100 REM LOW FREQUENCY NOTES * CALL SOUND MUST CONTAIN
    3 TONES AND A NOISE.  1ST TWO TONES MAY BE EITHER
    AUDIBLE OR INAUDIBLE.  3RD TONE MUST BE THE
110 REM FREQ CODE FOR NOTE WITH AN INAUDIBLE VOL AND
    THE NOISE MUST BE -4 WITH AN AUDIBLE VOLUME * PROGM
    BY JIM PETERSON MODIFIED BY DON MACCLELLAN
115 REM OPEN #1:"PIO"
120 CALL CLEAR
130 CALL SCREEN(11)
140 DEF R(X)=INT(X+.5)
150 F=1652
160 PRINT "  Note   Code   Frequency"::
165 REM PRINT #1:"  Note   Code  Frequency"
170 FOR BIGLP=1 TO 3
180 RESTORE 300
190 FOR LITLP=0 TO 11
200 READ N$
210 B$="2020303C223C0000"
220 C$="00"
230 CALL CHAR(37,B$)
240 CALL CHAR(39,C$)
250 PRINT "   ";N$;"   ";R(F);"  ";R(F)/15.01818182
255 REM PRINT #1:"   ";N$;"   ";R(F);" "R(F)/15.01818182
260 CALL SOUND(500,30000,30,30000,30,F,30,-4,0)
270 F=F/1.059463094
280 NEXT LITLP
290 NEXT BIGLP
295 CLOSE #1
270 DATA A',A%,G',F#,F',E',E%,D',C#,C',B',B%
```

The essence of the program, neglecting the graphics can be stated in one line in Extented Basic:

```
1 DEF R(X)=INT(X):: F=1652 :: FOR B=1 TO 3 :: FOR L=0 TO
  11 :: PRINT R(F);R(F)/15.0182 :: CALL SOUND(500,30000,
  30,30000,30,F,30,-4,0):: F=F/1.06 :: NEXT L :: NEXT B
```

PROGRAMS in DIS/VAR 80 FORMAT

by Rich Hubbard

Here's an 'interesting' tip for those with (or planning on acquiring) the TI-WRITER Word Processor. Have you noticed those programs printed in various publications in which all characters are aligned as they would be on a monitor when typed from the console? Here's how you can do it!

With your program loaded in console and a disk ready and waiting in your disk drive with sufficient space, type 'LIST "DSK1.filename"' and ENTER. (Suggest you use either a different disk or different filename than used with original program). Your program is now stored as a DIS/VAR 80 file which can be loaded and edited using the TI-WRITER software. After editing to meet your space requirements, print it using the FORMATTER. (Remember -- by editing I mean for printing only, not to change the run-time action of the program - this file cannot be recovered to 'RUN' as a program)

There must be some other applications for this 'LIST "DSK1.filename"' - if you know what they are, I'd appreciate your passing them on!

PRESSING FORTH

by John F. Schmidt

THE BOX PROGRAM

When I first purchased my TI-99/4A, it was with the idea that I could write a program which would be able to duplicate some of the behavior of the Atari Game 'Star Raiders', which I had seen in a store. I thought it was the most interesting game I had ever seen on a home computer. I did not try writing such a program right off the bat of course, but I endeavored to learn enough to soon do the job. One of my first major disappointments was the discovery that the graphics in "barefoot basic" were so primitive that continuous sprite motion was impossible. Also, there was no "PEEK" and "POKE' commands like the little Timex computer had. That meant that I had no direct access to the machine's memory. Later, I discovered that those commands would have been useless anyway, since TI had thoughtfully structured their computer's memory so that there was no read CPU memory to mess with anyway. (It's all in the video chip - not accessible unless you buy extra command cartridges. That way they could 'Command' a few more bucks from you.) Now I'm not trying to suggest that the 99/4A doesn't have sophisticated graphics capabilities, it's just that they aren't available for the average person like you and I. We normally don't have a degree in advanced programing, and most of us have no reliable relationship with greenbacks (except when waving goodbye to them as they are carted off by the IRS or bill collectors.) So to make a long story short, I had just about given up on my plan to write a TI version of Star Raiders when I heard about FORTH.

Ah, Rapture!! TI FORTH is to TI Basic what an M-16 is to a Water pistol. It is considerably different than Basic, and that perhaps explains why you may not know much about it. It has similarities to Basic, and that will help you learn it, and it has differences which will require some getting used to. Do you own a scientific calculator ? A Hewlett Packard perhaps ? If you do (I don't) you will find the method of operation a 'natural', since FORTH used the equivalent of 'Reverse Polish Notation'. (Really, that's what it's called) Reverse Polish Notation or 'RPN' for short, describes the method by which variables are entered to do a calculation. For example, suppose we want to add two numbers together, like 3 plus 5. The 'algebraic' method of operation requires the numbers to be entered like this: '3' '+' '5' '=' . The answer then appears. RPN requires this form: '3' '5' '+' That's all. The answer generally appears on the calculator display at that point. The difference is that with RPN, you push the numbers into a 'stack' format and with the 'algebraic' system you enter the numbers and commands in the sequence you would normally write them. In FORTH, to display a number from the stack onto the screen you type a '.' (period). That is a 'PRINT' statement in FORTH. Note that you must have whatever number you want to print already on the stack. It may be the result of a calculation, or it may be a number you just typed in. It doesn't matter. To put a number on the stack, type and (ENTER) it, or follow it with a blank and another number if you want more than one on the stack. The '.' command word prints off of the top of the stack, so it is like pushing and popping coins in and out of a spring loaded coin dispenser. The last coin in is the first coin out. Three periods in a row '. the stack, one by one, and print them to the screen. They won't be on the stack anymore when you finish. You 'spent' the coins.

Now that is just an idea of the way the stack works. To use the little program called "BOX", it is necessary to understand what the stack is, and a little about how it works. To use the box program you must specify the 'SPLIT' or 'SPLIT2' mode. That puts the computer in 'bitmap' mode. (That's the mode you must use to write a Star Raiders game, by the way) Bitmap allows you to separately define every single 'pixel' on the screen. By way of illustration, consider that a period (".") is four pixels in a square pattern. Pixels are the smallest mark the computer can make on the television screen or a printer. The box program draws a rectangle of length "LEN" and positions it's upper left corner at the coordinates 'DOT COLUMN', 'DOT ROW'. These two numbers locate a point on the screen as if it were a grid with each cell numbered. The row numbers start with the top row of one and the columns with the left column being number one. The upper left pixel then, is (1,1). The screen is 256 pixels wide and 192 pixels high. The lower right point on the screen is then (DROW,DCOL) = (192,256). By defining dots in a line, one after another, a line can be drawn on the screen. Now FORTH has thoughtfully taken care of the commands for a dot and a line. So it should not be very difficult to take the Line command word and use it repeatedly to make a square or "BOX". That is just what the little program does which is explained in HOLDING FORTH.

The command structure of FORTH is really very easy to use. Aside from the language being constructed around the concept of a stack, it is also built up from very simple commands called 'words'. Some languages call this process building a 'macro'. The users of the language can build his own set of special command words. To execute a word in FORTH, you just type it in and enter it. You might be wondering how one writes a program this way. It really is simple, and it forces you to construct the program in a systematic way called 'structured programming'. Now that is not so bad really....It's just good thinking. Beyond that, structured programs are very easy to troubleshoot, since their logic is so simple to follow. To program with FORTH, you begin by analyzing the task and naming it by some word. This word must be defined as a series of other words. These other words in turn, accomplish the series of steps necessary to do the function desired. It is sometimes necessary for each of the first words to themselves be broken down into other simpler words, and so on. In this way, the problem is broken into managable pieces. Each 'piece' or word can be separately tested also, so that debugging becomes much simpler since the components of the proram have already been tested.

*** HOLDING FORTH ***

by John F. Schmidt
* A Column on the TI-FORTH Language *

This column is devoted to those who are interested in making their TI-99/ do more than they ever dreamed it could do using Basic. To use the TI-FORTH language it will be necessary to have at a minimum a disc drive system with the memory expansion system. If you don't have that, I suggest that you get in contact with some club member who does, and work with him, or get involved in one of the informal FORTH interest groups which are springing up. The article which follows describes a command word which will draw a 'BOX' or square anywhere on the screen in Bit-Map mode. If you are interested in Bit-Map graphics of any kind, FORTH is definitely for you. It is easy to learn and is vastly more powerful than Basic, and accesses all of the resources of the TI-99/4A without a lot of fuss.

Here is a description of how to use the 'BOX' routine written for the TI-FORTH language. The program BOX uses the bit-mapped mode of screen display so that the programmer has the highest resolution available to him. Either 'SPLIT' or 'SPLIT2' mode can be used, although one must assure that the row and column chosen fits the active portion of the screen which is available.

The Box word uses the already-defined FORTH word 'LINE' four times in order to make a box or square. The input format required is three numbers: Dotcolumn, Dotrow, Dotlength of one side. The Drow, Dcol numbers locate the upper left corner of a square with a side of length 'LEN'. It is necessary to push these numbers into the stack before calling the word. An example would be a box located near the center of the screen. The command would be entered as " 128 98 25 BOX " .

Here's how it is done (and this certainly isn't the last word on how!). Line 2 tells the computer to save the return address so the computer can return to what it was doing before we called the definition. The word 'DECIMAL' tells the computer to regard all numbers you give it as decimal, as opposed to hexidecimal or binary or whatever.

Line 3 defines three variables and puts zeros into them. These are the Length, the Dotrow and the Dotcolumn. We will need these after we get them off of the stack.

Before we discuss lines 4 through 7 let's look at the main driver program that is found in lines 8 to 13. Notice that the first thing we find in line 8 is a colon (:). That tells the computer that we are going to define a new word. It's name is whatever follows the colon; in this case, "BOX". Until the computer finds a semicolon (;), it will regard all subsequent numbers, words, etc as belonging to the definition of the operation of the word "BOX".

Line 9 calls our variable "LEN", and puts it's address on the stack. The " ! " sign, which is actually a word, tells the computer to take the number below 'LEN' on the stack and put it into the memory location assigned to 'LEN'. Remember that the last number we put on the stack was the length. That is at the top of the stack. (The next one down is Drow, then Dcol on the bottom). The rest of line 9 repeats the same type of operation described for LEN for the variables 'DROW' and 'DCOL'. When line 9 finishes executing, the data we put on the stack before we called 'BOX' is now in three vaariables called LEN, DROW, and DCOL. Once we have them defined, we can use them over and over, without losing them like we would if we took them directly off of the stack when we needed them. This is like "LET LEN=123" in Basic.

In line 10 we see the use of the word 'LINE'. This a 'System' word and is there for us to use. It requires us to tell it the Dotcol and Dotrow of one end of a line, and the Dotcol and Dotrow of the other end of the line; it draws the line for us when we call it. That's pretty handy for us. Line 10 uses two words "DP1" and "DP2" to do our book keeping for us. DP1 puts the values for Dotposition 1 onto the stack, and DP2 does the same for Dotposition 2. See how it becomes easy to make up words which contain complex instructions ? If you will visualize the box corners as numbered 1 through 4 starting at the upper left and proceeding clockwise, then the lines 10 through 13 become easy to read and understand. Notice that line 13 ends with a semicolon (;). Remember why ?

Lines 4 through 7 define the words DP1,2,3 and DP4. These take the data we saved in 'LEN', 'DCOL' and 'DROW' and calculate the correct dot and row positions for us to use in lines 10 through 13. Let us look at two representative samples of these to see how they work. Line 4 describes DP1 and DCOL+. DP1 takes the address of our variable DCOL and pushes it onto the stack. The " @ " sign (again, another command word) instructs the computer to take whatever is stored at the address of Dcol and put that number onto the stack in place of the address. (The word " @ " does just the opposite of " ! " .) The next word combination does the same for Drow, so that when we encounter the semicolon after the second " @ " on line 4 we have put the value for Dcol onto the stack. To repeat: The calling of DP1 word puts the dot column value onto the stack, then puts the dot row value onto the stack. When it has done that it is finished, and returns to where it was called. The word DCOL+ is a special for of DCOL. Recall that the corners of the box are defined starting at the upper left corner, and the length is given as LEN. From this information, ti is easy to define the other corners. For instance, the #2 corner (upper right) can be defined as (DCOL + Len), (DROW). That is, the row number hasn't changed at all, only we have moved over from Dcol to Dcol plus LEN. Using this logic, we can see how DCOL+ works. Look at line 4 again. The definition for DCOL+ begins after the semicolon ending the definition for DP1. The colon starts the new definition.

The first command word following the name of the routine is DCOL. Remember that when a variable name is stated like this in FORTH, the meaning is to put the address of the variable on the stack....(not the value of the variable). The " @ " sign, a command word in its own right, tells the computer to take the value stored at the address found on the top of the stack, and replace the address with the value on the stack. So the combination of 'DCOL @' puts the DCOL value on the stack. The combination 'LEN @' put the value for length on the stack next. The ' + ' sign following these is a command to get the top two values off of the stack and add them together, and put the result back onto the stack again. The semicolon follows, since we have accomplished the desired result: To create a new DCOL value increased by the amount 'LEN'.

If you will study line 5, you will notice that essentially, we have just repeated the same kind of operation for the other dot and column locations. When we are finished, we have defined words which give coordinate values for each of the corners of the box. All that would remain to do is to use these to call the line routine four times, using the appropriate coordinate words. We have already seen that done in lines 10 - 13.

The last line is just the reset of the return address, which is the opposite of what was done in line 2. Note that your computer will stay in decimal mode unless you change it coming out of this screen. It is good practice to set the base of the number system on entering a word definition.

Keep on PRESSING FORTH while I continue HOLDING FORTH !

```
SCR #30
 0  ( BOX ROUTINE.  ENTER DCOL,DROW,LEN THEN 'BOX' )
 1
 2 BASE->R DECIMAL
 3 0 VARIABLE LEN 0 VARIABLE DROW 0 VARIABLE DCOL
 4 : DP1 DCOL @ DROW @ ;  : DCOL+ DCOL @ LEN @ + ;
 5 : DP2 DCOL+ DROW @ ;  : DCOL+ DROW @ LEN @ + ;
 6 : DP3 DCOL+ DROW+  ;
 7 : DP4 DCOL @ DROW+ ;
 8 : BOX
 9   LEN !  DROW !  DCOL !
10   DP1 DP2 LINE
11   DP2 DP3 LINE
12   DP3 DP4 LINE
13   DP4 DP1 LINE  ;
14
15 R->BASE
```

NEW MEMBERS WELCOME

The BYTEMONGER is being distributed to a number of Lexington Computer hardware and software suppliers this month in an attempt to reach many TI-99/4A owners who may not know of our orgainzation. If you are one of those, annual membership is $12.00 which may be sent to our P.O. Box 11866 if you are unable to attend this month's meeting.

* LIBRARY UPDATE *

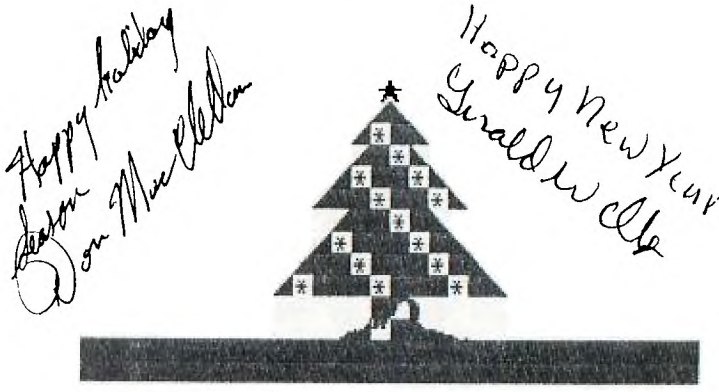
There will be several new public domain programs available in the library at the December meeting. There will be only one copy of each diskette in the library for loan, but extra copies of each will be available which you may have with a replacement diskette or replacement cost. Among the offerings will be a diskette with four programs which, by the use of key words, find the issue and page number of articles and programs in magazines. The search programs are for COMPUTE!, HOME COMPUTER MAGAZINE, FAMILY COMPUTING MAGAZINE, and the book 101 PROGRAMMING TIPS & TRICKS for the TI-99/4A. The search programs for 101 TIPS & TRICKS and COMPUTE! will also be available on cassette. The book 101 PROGRAMING TIPS has also been added to the library. Because of the length of the HCM program, it is not possible to put it on cassette. We would welcome a volunteers help to separate the index by year or volume so that it may be put on several cassette tapes. If you want to attempt this small task, please bring a diskette to the meeting in order to get a work-copy of the program.

FORTH & ASSEMBLY LANGUAGE NATIONAL LIBRARY

The 99'ers Users Group Association in Bakersfield, California is in the process of establishing a much needed *library for the accumulation* and distribution of FORTH and ASSEMBLY Language programs and they have issued a request for programs which can be dedicated to the Public Domain in both FORTH and ASSEMBLY language. While there is a wealth of potential for utilizing the power of the TI-99/4A machine's capability in both of these powerful languages, there has not been in the past a means of contributing and sharing in the programs which have been written. This project is a much needed service. It is important that we support this program with some contributions if we expect to gain from accessing the many programs which should be made available through the Library's distribution.

It is significant to note that, while the TI-99/4A is no longer manufactured or supported by Texas Instruments hardware and software development, there seems to be more programs available today from third party developers than were available when the unit was being produced. This should be reassuring to all TI-99/4A owners that the machine, while it has some designed in obstacles to third party suppliers, is still a powerful and useful computer which will become obsolete only when service is impossible to obtain.

See any of the Group officers for the proper forms to be used in submitting programs for the Library.....Ed.

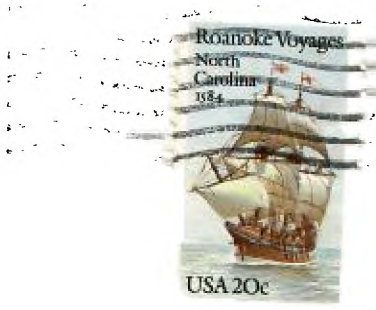


MERRY CHRISTMAS

FROM ALL THE MONGERS

BYTE BIT NIBBLES AND LITTLE NIBBLET

BLUEGRASS AREA 99er USERS GROUP
P. O. BOX 11866
LEXINGTON, KENTUCKY 40578-1866



EDMONTON 99'er CU Soc
P.O.Box 11763
Edmonton Alberta Canada T5J 3L1