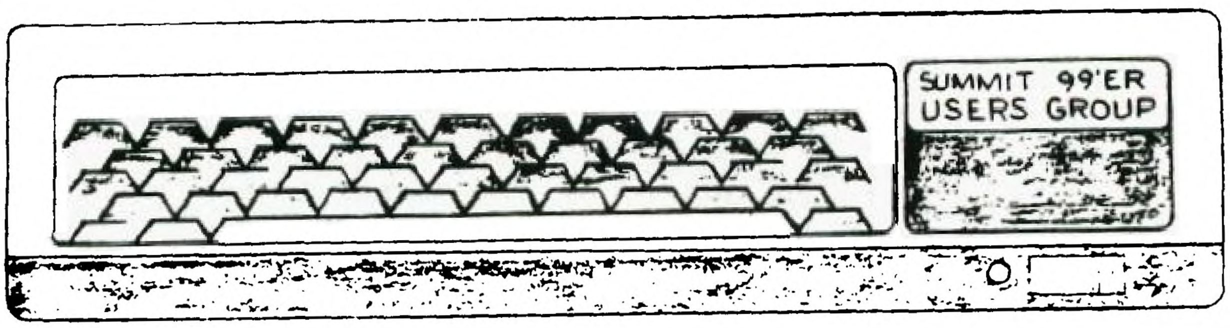

DECEMBER 1984 Vol. 2 No. 12

The December meeting will be held on Thursday. December 20 at Cuyahoga Falls High School at the corner of Fourth and Stow streets in Room 413 - Physic's Lab. Please remember to sign in. The January meeting will be held on January 17th.

This month we will have a Christmas party for our December program. We are asking that those attending this month's meeting bring 1 dozen cookies. We will supply coffee and punch.

SWAP MEETING

You can bring your blank tapes and copy tapes from our library or any public domain program. It would be wise to bring your cassette recorder if not your entire system to take advantage of our library.

BASIC CLASS

Rich's subject this month will be "How to Load and Save on Cassette". Bring your unit: monitor, keyboard, cassette recorder, and blank tape. He will also have a Question and Answer period on all levels of Basic.

MEMBERSHIP DUES

Those people whose membership dues are due in January will be due in January. Remember the constitution voted on this year, increased the dues to $15.00 from $10.00 last year.

PRESIDENTS CORNER

I want to start by wishing you all a very Merry Christmas and a Happy New Year !

Christmas to me, is a very special time of the year, not because of all of the presents, parties and fattening food. (they are all great and I love them) But I think of all the loved ones and friends I have, and just how much they all mean to me. In the time I've been in this group, I've come to think of most of you as friends, for this I am thankful.

I have some good news, it looks as though we will continue to meet at the Cuyahoga Falls High School as we did for the last year. But with this major differance, we will be insured and leagal with a contract for one year, from the school.

The comming year looks to be quite interesting. The companies that expected to make a killing off of the computer boom that was going to sweep the country. And so called experts, analists, and Madison Ave. Pitchmen, have had the wind knocked out of their sails.

They forgot one very important variable. People! Some people, and the world as a whole are not ready to have a computer in every home, or access to a Data Base at their finger tips, no matter where they are at the time.

One good thing did come of this rush and rude awakening, cheap computers! This lead many who otherwise would not have even thought of using a computer, to buy one if not two or three.

Many people have them stored away with their C.B.'s and camera equipment, waiting until they have enough time to figure out all of the technical stuff. While others have just sold out." It's just not for me". " It costs too much to get software". " I can't get the hang of programing" - or - the classic " The manufacturer got out and stuck me with something I can't use". Any excuse is better than none.

There are also a few who have heard all of the excuses and sad stories, they could become true computer lovers if they take the time to think, instead of feeling that changing from one brand to another, will cure all of the problems that the first computer didn't.

This all leads us to the point of my story. After all of the shouting and scrambling, the ones that are left will find that the wait was worth it and that the future is bright. Like anything else with a little information, and suport from friends our fears fade away.

At our last meeting we had a visitor, named Larry Moyer and his family. He told us of a mail order T.I. supplier. the name and address is EDU-COMP 6516 O'HENRY CIRCLE N. RIDGEVILLE, OHIO 44039 PHONE (216) 327-6579 . The person that I contacted is called Judy Thalner. She pretty much runs a one person show. the speed and professionalisim that I was shown, along with the price and speed of the processing of my order was very impressive.

Again MERRY CHRISTMAS and try to make our party.

NORM SORKIN

TIPS FROM THE TIGERCUB

#16

TIGERCUB SOFTWARE
156 Collingwood Ave.,
Columbus OH 43213

These Tips are being mailed, together with my new catalog #5, to every Users Group that I know of. I hope that you will make both the Tips and the catalog available to your membership. I am sorry that I cannot take out paid ads in your newsletters, but to advertise in each one of them would cost me more than I have made in the past 6 months, and I would not get enough business to break even.

If you would like to continue receiving these Tips, put me on the mailing list for your newsletter, and give me some indication that my Tips are really reaching your members and not going into someone's private file. If I receive enough business from this mailing to pay for its cost, I will then continue to send you my Tips. If not, this will be the last issue of the Tips from the Tigercub.

Copies of my catalog are available for $1.00, which is deductable from your first order. I have over 130 absolutely original quality programs in Basic, many of them now also available in XBasic, on casette or disk for only $3.00 each plus $1.50 per order for casette, package and postage, or $3.00 for diskette, package and postage (higher overseas). I give one-day service, I give bonuses for repeat orders, I give bonus programs on diskette orders.

In addition, any User's Group member who mentions his/her users'group when sending me an order before 1 Jan. 1985 may deduct 10% from the cost of the programs.

Tips from the Tigercub #1 thru #14 are now available, with more added, as a diskfull of 50 programs, routines and files for only $15 postpaid.

I have also now completed my NUTS & BOLTS disk of 100 XBasic utility subprograms in MERGE format, ready to merge into your own programs, for just $19.95 postpaid.

In The last Tips, I mentioned that I wished I knew who to credit for that remarkable routine to redefine the cursor. Dave Peden has written me that credit should be given to Terry L. Atkinson of 28 Savona Ct., Dartmouth, NS B2W 4R1 CANADA.

And I would like to strongly recommend that you support the 99'ers Users Group Association, 3535 So. H st., #93, Bakersfield CA 93304. They are a strictly non-profit group, devoting a lot of time and effort to helping us all, and they publish a great newsletter..

Every Tips must include a bit of music, and my grandson has requested that I pass this one on to all other two-year olds.

```
100 !ALPHABET SONG - by Jim
Peterson
110 DIM N(21)
120 CALL MAJORSCALE("C",N())
130 CALL SCREEN(5):: DISPLAY
 AT(24,1)ERASE ALL:"READY -
TYPE THE ALPHABET" :: CALL M
AGNIFY(2)
140 CALL KEY(3,K,ST):: IF (S
T<1)+(K<65)+(K>90)THEN 140 :
: CALL SPRITE(#1,K,16,96,120
):: IF K=87 THEN GOSUB 220 E
LSE GOSUB 200
150 IF (K=90)*(FLAG=0)THEN 1
60 ELSE 140
160 FLAG=1 :: M$="C115566D5C
443322D1" :: T=150
165 FOR J=1 TO 18 :: CALL SP
RITE(#J,64+J,INT(11*RND+6),9
6,128,J*5,J*5)
170 X=ASC(SEG$(M$,J,1)):: IF
 X>58 THEN T=150*(X-64):: GO
TO 190
180 X=X-48 :: CALL SOUND(T,N
(X),0)
190 NEXT J :: FLAG=0 :: CALL
 DELSPRITE(ALL):: GOTO 140
200 Y=VAL(SEG$("115566544332
22215543325332",K-64,1))
210 CALL SOUND(500,N(Y),0)::
 RETURN
220 CALL SOUND(500,N(5),0)::
 CALL SOUND(500,N(5),5):: CA
LL SOUND(500,N(4),0):: RETUR
N
230 SUB MAJORSCALE(K$,N())
240 F=VAL(SEG$("110123131147
165175196",POS("ABCDEFG",K$,
1)*3-2,3))
250 C$="10101101010110101101
0101101011010101"
260 FOR J=1 TO 36 :: IF SEG$
(C$,J,1)="0" THEN 280
270 X=X+1 :: N(X)=F*1.059463
094^(J-1)
280 NEXT J :: SUBEND
```

Lines 230-280 of that routine are an example of the kind of handy-dandy subprograms you will find on my Nuts & Bolts disk.

We haven't had a Tigercub Challenge for some time, so -

How can you store a hundred or more values of any size, positive or negative, integer or non-integer, even in exponential notation, without dimensioning an array or opening a file?

Now, how can you link your program to another by a RUN statement, thereby losing all data, and recover those values? Yes, I know you can save them on the screen and read them back, but can you find a better way?

Here's a little demo program of how motion can be created by the repetitive redefinition of characters. I call it ETERNITY.

```
100 CALL CLEAR :: CALL SCREE
N(2):: CALL COLOR(1,16,1)::
CALL CHAR(33,"",34,"",35,"",
36,"")
120 FOR R=1 TO 12 :: CALL HC
```

```
HAR(R,R+4,33,26-R*2):: NEXT
R
150 FOR R=13 TO 24 :: CALL H
CHAR(R,29-R,34,(R-12)*2):: N
EXT R
160 FOR C=5 TO 16 :: CALL VC
HAR(C-4,C,35,34-C*2):: NEXT
C
210 FOR C=17 TO 28 :: CALL V
CHAR(29-C,C,36,C*2-33):: NEX
T C
225 FOR J=0 TO 7 :: A$(J+1),
B$(8-J)=SEG$("00000000000000
",1,2*J)&"FF" :: NEXT J
230 C$(1),D$(8)=RPT$("80",8)
:: C$(2),D$(7)=RPT$("40",8):
: C$(3),D$(6)=RPT$("20",8)::
 C$(4),D$(5)=RPT$("10",8)
240 C$(5),D$(4)=RPT$("08",8)
:: C$(6),D$(3)=RPT$("04",8):
: C$(7),D$(2)=RPT$("02",8)::
 C$(8),D$(1)=RPT$("81",8)
250 FOR C=2 TO 15 :: FOR J=1
 TO 8 :: CALL CHAR(33,A$(J),
34,B$(J),35,C$(J),36,D$(J)):
: NEXT J :: CALL SCREEN(C)::
 NEXT C :: GOTO 250
```

Next, I would like to share with you a gem of a "why didn't I think of that" routine which John Taylor sent me.

```
100 ! 28 COLUMN TEXT ROUTINE
IN EXTENDED BASIC (EASILY
CONVERTED TO BASIC) BY JULIE
PACK, B.U.G., P.O. BOX 1402
PALM BAY, FL 32906
110 ! ENHANCED BY JET
SHOALS 99'ERS, P.O. BOX 2928
MUSCLE SHOALS, AL 35662
120 CALL CHAR(64,"00282828")
130 ! PROGRAM TO COPY STARTS
     HERE
140 CALL CLEAR :: X=-1
150 RESTORE
160 IF X>=21 THEN X=1 :: CAL
L WAIT
170 READ MESS$
180 IF MESS$="P" THEN DISPLA
Y AT(X+2,1):Z$ :: X=X+4 :: Z
$="" :: GOTO 160
190 IF MESS$="ZZZ" THEN DISP
LAY AT(X+2,1):Z$ :: CALL WAI
T :: END
200 IF LEN(Z$)>0 THEN MESS$=
Z$&" "&MESS$
210 X=X+2
220 IF X>=21 THEN X=1 :: CAL
L WAIT
230 IF LEN(MESS$)<29 THEN DI
SPLAY AT(X,1):MESS$ :: Z$=""
 :: GOTO 160
240 FOR A=1 TO 29
250 I=POS(MESS$," ",A)
260 IF (I=0 OR I>29)AND A=1
THEN A,J=29 :: GOTO 290
270 IF I=0 OR I>29 THEN A=29
 :: GOTO 290
280 J,A=I
290 NEXT A
300 IF X>=21 THEN DISPLAY AT
(X,1):SEG$(MESS$,1,J-1):: X=
-1 :: CALL WAIT :: GOTO 320
310 DISPLAY AT(X,1):SEG$(MES
S$,1,J-1)
320 IF SEG$(MESS$,J,1)=" " T
HEN I=1 ELSE I=0
330 Z$=SEG$(MESS$,J+I,163)::
 MESS$=Z$ :: IF LEN(Z$)>28 T
HEN X=X+2 :: GOTO 240
340 GOTO 160
350 DATA "THIS SHORT ROUTINE
 WILL ENABLE YOU TO WRITE LO
NG TEXT MATERIAL IN YOUR DAT
A STATEMENTS SO YOU WON'T HA
VE TO WORRY ABOUT COUNTING"
360 DATA "THE LENGTH OF YOUR
 SENTENCES ALL THE TIME.  TH
IS ROUTINE WILL AUTOMATICALL
Y EDIT YOUR TEXT TO FIT A 28
 COLUMN SCREEN."
370 DATA "A SUGGESTION- IT I
S A GOOD IDEA TO PUT A QUOTE
 AT THE BEGINNING AND END OF
 THE DATA STATEMENTS SO YOU
WON'T HAVE TO WORRY ABOUT"
380 DATA "COMMAS LIKE THIS ,
,, AND THEY WILL REMAIN IN Y
OUR TEXT PROPERLY."
390 DATA "THIS ROUTINE WILL
ALSO CLEAR THE SCREEN (WHEN
FILLED) AND CONTINUE READING
 YOUR DATA AND DISPLAYING YO
UR TEXT ON THE NEXT SCREEN."
400 DATA P
410 DATA "  TO START A NEW P
ARAGRAPH ENTER THE LETTER @P
@ AS A SEPERATE DATA STATEME
NT, THEN INDENT YOUR TEXT ON
 YOUR NEXT NEXT DATA"
420 DATA "STATEMENT 2 OR 3 S
PACES (IF DESIRED).",P,"TO S
KIP LINES,",P,"JUST ENTER @P
@",P,"WHERE EVER YOU WANT TO
",P,"SKIP."
430 DATA P,"MAKE SURE THAT Y
OUR VERY LAST DATA STATEMENT
 IS @ZZZ@, AND JUST REPLACE
THESE DATA STATEMENTS WITH"
440 DATA "YOUR OWN.",P,"YOU'
LL ALSO FIND THIS ROUTINE IS
 MOST USEFUL WHEN CONCATENAT
ING STRINGS, E.G., @ELIZA@ T
YPE PROGRAMS-",P
450 DATA "AN EXAMPLE:",P,"A$
=@JACK AND JILL WENT UP@","B
$=@THE HILL TO FETCH A@","C$
=@PAIL OF WATER.@","D$=A$&B$
&C$&D$","PRINT D$",P
460 DATA "JACK AND JILL WENT
 UP THE HILL TO FETCH A PAIL
 OF WATER.",P,P,P,"HAPPY PRO
GRAMMING!"
470 DATA ZZZ
480 SUB WAIT
490 DISPLAY AT(24,8):"PRESS
ANY KEY"
500 CALL KEY(0,K,S):: IF S=0
 THEN 500 ELSE CALL CLEAR
510 SUBEND
```

Thank you, Julie and John. This is becoming one of the most useful routines on my utility disk. I was preparing a disk of PD programs for our UG library. Some of them needed extra instructions, so I typed them out on TI-Writer, so that people could run them off on their printer. Then I remembered that some folks don't have printers. So -

```
50 CALL CLEAR :: INPUT "FILE
NAME? DSK1.":F$
60 DIM B$(150):: OPEN #1:"DS
K1."&F$,INPUT, DISPLAY ,VAR
IABLE 80
70 A=A+1 :: LINPUT #1:B$(A)
80 IF EOF(1)=1 THEN B$(A+1)=
"ZZZ" ELSE 70
```

and change line 170 to -

```
170 @=@+1 :: MESS$=B$(@)
```

And there you have a quickie program to check out those DIS/VAR 80 files that show up on your disks under filenames that you can't remember using.

MEMORY FULL IN LINE 32767

[illegible]

This article on Forth comes to us from BYTEMONGER in Lexington, KY. December 1984 issue.

*** 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
```

Beginning November 5, 1984

CHANNELS 45/49

A 12-WEEK T.V. SERIES, BITS AND BYTES

MONDAYS AT NOON

THURSDAYS AT 10:30 PM

SATURDAYS AT 6:30 PM

45 WNEO
49 WEAO

An exciting new way to learn about computers

Television Series made possible by grants from Computerland and Second National Bank of Warren.

ACADEMY ON COMPUTERS

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.

TIPS FROM THE TIGERCUB

#17

Copyright 1984

TIGERCUB SOFTWARE
156 Collingwood Ave.,
Columbus OH 43213

Distributed by Tigercub Software to TI-99/4A Users Groups for promotional purposes and in exchange for their newsletters. May be reprinted by non-profit Users' Groups, with credit to Tigercub Software.

My new catalog #5 is now available for $1.00, which is deductable from your first order. It contains over 130 programs in Basic and Extended Basic at only $3.00 each (plus $1.50 per order for casette, packing and postage, or $3.00 for diskette, PP&M).

The entire contents of Tips from the Tigercub Nos. 1 through 14, with more added, are now available as a full disk of 50 programs, routines and files for only $15.00 postpaid.

Nuts & Bolts is a diskfull of 100 (that's right, 100!) XBasic utility subprograms in MERGE format, ready for you to merge into your own programs. Contents include 13 type fonts, 14 text display routines, 12 sorts and shuffles, 9 data saving and reading routines, 9 wipes, 8 pauses, 5 music, 2 protection, etc., etc., all for just $19.95 postpaid!

And if you send an order before 31 December 1984 and mention your user group, you may take a 10% discount.

My 28-Column Converter, published in Tips #15, has a bug which causes a line to disappear if the wrap-around causes it to begin with a period and you are using the formatter option. Here is the fix -

Change line 300 to read: 300 FOR W=1 TO 5 :: READ CH$,R$

Change line 280 to read:

280 DATA @,(,&,),^,~,*,!,.,\ In other words, your DATA items will be the "at" sign above the 2, the left brace on the front of the F key, the ampersand on the 7 key, the right brace on the front of the G, the carat sign above the 6, the tilde on the front of the W, the asterisk above the 8, the whatsit? on the front of the A, the period, and the backslash on the front of the Z.

A couple of other changes will automatically turn off the automatic fill and adjust, and turn it back on. At the end of line 180, add :: PRINT #2:".NF" and change line 270 to NEXT J :: PRINT #2:".FI;AD;"

:: CLOSE #2 :: CLOSE #1 :: END

Now, as long as the text strings in your program don't contain those oddball characters, all should be well. However, the program has one more bug which is common to all 28-column converter programs, and for which I can find no really good fix. If a program line is exactly 60 characters long, the next program line will follow immediately after it instead of starting on the next line. So, load the file in the Editor mode and scan it before you print it. If any of you whiz kids (or whiz grandpas) can figure out a way to program around that problem, please let me know!

A challenge in Tips #9 was to write a 1-line XBasic program which would take only 70 seconds to scramble the numbers from 1 to 255 into a completely random sequence without duplication. Richard Mitchell, the editor of Super 99 Monthly, came up with an algorithm which is shorter than mine and runs about 10 seconds faster - but it sure does chew up a lot of memory!

```
1 DIM A(255),C(254):: RANDOM
IZE :: CALL PEEK(-31808,B)::
IF B=0 OR A(B)=B THEN 1 ELS
E C(D)=B :: A(B)=B :: D=D+1
:: IF D=255 THEN END ELSE 1
```

And if you're not subscribing to Super 99 Monthly, you should be! It's only $12 a year, and full of very useful programs, routines and tips. The address is Bytemaster Computer Services, 171 Mustang Street, Sulphur LA 70663.

Also be sure to get the National Ninety-Niner from the 99ers Users Group Association (3535 So. H St. #93, Bakersfield CA 93304), also only $12 a year. Their roster of writers is beginning to look like the Who's Who of the TI world.

Danny Michael has written an assembly language program which will dump a graphics screen to a dot matrix printer (Epson or Gemini, and probably others) in less than 50 seconds - and he's giving it away. Just send him an initialized disk in a diskette mailer with an address label back to you and enough return postage. His address is Route 9, Box 460, Florence AL 35630.

Please, can ANYONE tell me where can buy diskette mailers at a decent price? The cheapest I have found are $0.65 each for an 11" x 9" piece of cardboard!

Somebody said they liked my Alphabet Song in the last Tips, and somebody else wanted some more routines for the speech synthesizer, so I put it all together and here's what I came up with. If you can type the alphabet without a mistake, you get an encore.

```
100 CALL CLEAR
110 PRINT "       ALPHABET S
ONG"
120 FOR J=1 TO 20
130 PRINT
140 NEXT J
150 PRINT "              by Ji
m Peterson": :"Wait, please"
;
160 OPEN #1:"SPEECH",OUTPUT
170 DIM T$(26,2)
180 DATA 12,12,4,4,1,1,4,7,7
,8,8,10,10,10,10,12,4,4,7,8,
6,10,4,8,8,10
190 FOR J=1 TO 26
200 READ X
210 T$(J,1)="//"&STR$(X)&" "
&STR$(X/10*32)
220 T$(J,2)=CHR$(J+64)
230 NEXT J
240 T$(23,2)="DOUBLE"&"!"&"!
"&"U"
250 CALL CLEAR
260 PRINT "READY - TYPE THE
```

```
ALPHABET"
270 T=0
280 K2=64
290 CALL KEY(3,K,ST)
300 IF (ST<1)+(K<65)+(K>90)T
HEN 290
310 IF K<>K2+1 THEN 330
320 T=T+1
330 PRINT #1:T$(K-64,1):T$(K
-64,2)
340 CALL HCHAR(12,17,K)
350 K2=K
360 IF K<>90 THEN 290
370 IF T=26 THEN 390
380 GOTO 270
390 FOR K=65 TO 90
400 CALL HCHAR(12,17,K)
410 PRINT #1:T$(K-64,1):T$(K
-64,2)
420 NEXT K
430 PRINT #1:T$(1,1):"NOW IV
E":T$(3,1):"SAID MY":T$(5,1)
:"A B":T$(3,1):"SEEZ"
440 PRINT #1:T$(8,1):"WONT Y
OU":T$(10,1):"COME AND":T$(1
2,1):"PLAY WITH":T$(1,1):"ME
"
450 GOTO 270
```

Terry Atkinson's routine to redefine the cursor has aroused some interest, so I fiddled around and came up with this version to change the cursor automatically to whatever character, normal or redefined, that you input.

```
100 !CURSOR CHANGER by Jim P
eterson
110 INPUT A$ :: A=ASC(A$)::
CALL CHARPAT(A,A$):: FOR J=1
 TO 16 STEP 2 :: H$=SEG$(A$,
J,2):: CALL HEX_DEC(H$,D)::
T=T+1 :: H(T)=D :: NEXT J ::
120 CALL INIT :: CALL LOAD(8
196,63,248)
130 CALL LOAD(16376,67,85,82
,83,79,82,48,8)
140 CALL LOAD(12288,H(1),H(2
),H(3),H(4),H(5),H(6),H(7),H
(8))
150 CALL LOAD(12296,2,0,3,24
0,2,1,48,0,2,2,0,8,4,32,32,3
6,4,91)
160 CALL LINK("CURSOR")!THAN
KS TO TERRY ATKINSON
170 SUB HEX_DEC(H$,D):: N=1
:: DEC=0
180 FOR J=1 TO LEN(H$):: A$=
SEG$(H$,LEN(H$)-J+1,1):: IF
ASC(A$)>58 THEN HT=ASC(A$)-5
5 ELSE HT=VAL(A$)
190 DEC=DEC+N*HT :: N=N*16 :
: NEXT J
200 IF DEC<>32768 THEN D=DEC
 ELSE D=-(65536-DEC)
210 SUBEND
```

And of course you can always color the cursor with CALL COLOR(0,5,11) or whatever colors you like.

Most folks don't seem to know, and some folks refuse to believe, that the Memory Expansion can't store strings. If you are one of the disbelievers, plug in your Memory Expansion and try this -

```
100 FOR J=1 TO 255 :: M$=M$&
CHR$(J):: NEXT J
110 DIM A$(100):: X=X+1 :: A
$(X)=M$ :: PRINT X :: GOTO 1
10
```

Now RUN that. On my console, I get MEMORY FULL when X=43 although the SIZE command shows I have 24399 bytes of program space free (in the Expansion) - but only 204 bytes of free stack (in the console). Without the Memory Expansion I can get X up to 51, and in Basic to 53.

This can be a serious handicap if you are running a program which reads in a large number of strings from DATA statements, or generates strings while running.

Of course, when the Memory Expansion is attached, the program and the numeric variables are stored in the Expansion, leaving all the console memory available for strings - but if you do not generate strings, the console memory remains unused, because numeric data cannot overflow into it!

If your program generates more numeric variables than the Memory Expansion can hold, you can however store them in the console by converting them to strings, using STR$, and convert them back to numbers with VAL. This will allow you store an additional 700 to 900 or more numbers. Try this -

```
100 DIM A(3040),A$(1000):: F
OR X=1 TO 3000 :: A(X)=99 ::
PRINT X :: NEXT X
110 Y=Y+1 :: A$(Y)=STR$(99)
:: PRINT Y :: GOTO 110
```

When you get MEMORY FULL, type SIZE.

Dave Renkenberger sent me a neat little routine, and I played around with it a bit. For you who are not football fans, I'd better explain that the Wave is performed at football stadiums when the cheerleaders get the fans to stand and cheer, one seating section at a time, across the stadium - and those drunks on the roof are usually out of sequence.

```
90 !THE WAVE by David Renken
berger/modified by Jim Peter
son
100 CALL CLEAR :: CALL SCREE
N(4)
110 A$="**the wave**"
120 DISPLAY AT(4,14-LEN(A$)/
2):A$
130 E$="press any key to sto
p"
140 DISPLAY AT(22,14-LEN(E$)
72):B$
150 B$="995A3C3C3C3C2466"
160 A$="0000181B7EBD3C3C"
170 FOR CH=91 TO 118 :: CALL
 CHAR(CH,A$):: M$=M$&CHR$(CH
):: NEXT CH :: FOR R=8 TO 12
 :: DISPLAY AT(R,1):M$ :: NE
XT R
175 FOR T=1 TO 26 STEP 5 ::
DISPLAY AT(22,T):SEG$(M$,T,1
):: NEXT T
180 FOR CH=91 TO 123 :: CALL
 CHAR(CH,B$):: CALL CHAR(CH-
5,A$):: CALL SOUND(-999,-7,5
*RND):: CALL KEY(3,K,ST):: I
F ST<>0 THEN STOP
190 NEXT CH :: GOTO 180
```

MEMORY FULL

Happy hackin'

Jim Peterson

AKRON OHIO 1984

SUMMIT 99'er USERS GROUP
Kathi Anderson, Editor
P.O. Box 3201
Cuyahoga Falls, OH 44223

COMING MEETINGS

January meeting	January 17	Modems
Board meeting	January 24	
February meeting	February 21	Forth Demo
Board meeting	February 28	
March meeting	March 21	
Board meeting	March 28	

NEWSLETTER DEADLINE

The deadline for the January newsletter is January 5.

Walter Mott has for sale a new Mini-Memory for $65.00 or Logo II. You can reach him at 724-7240.

Everyone on the board would like to wish you and yours the best wishes for this holiday season. Hope this coming year is full of happiness for all our members and the Users groups across the country. Come and join us at the coming meeting and help us celebrate all of our good fortunes.

Kathi Anderson, Editor