THE GUILFORD 99'ER NEWSLETTER

VOL.6 NO.8 AUGUST 1989

The Guilford 99'er Users' Group Newsletter is free to dues paying members (One copy per family, please). Dues are $12.00 per family, per year. Send check to 3202 Canterbury Dr., Greensboro, NC 27400. The Software Library is for dues paying members only. (George von Seth, Ed.:292-2035)

OUR NEXT MEETING

DATE: August 1, 1989 Time: 7:30 PM. Place: Glenwood Recreation Center, 2010 S. Chapman Street.

Program for this meeting will be a demonstration of TI-BASE by Tony Kleen. Be sure to attend and see this dynamite data-base in action!!

MINUTES

The monthly meeting of the Guilford 99er Users' Group met on Wednesday, July 5th at the Glenwood Recreation Center on Chapman St. in Greensboro, N.C. There were 7 members present.

President Scott Hughes opened the meeting at 7:39 P.M. The minutes of the June meeting were read and accepted as read.

Old Business:

a. Scott lead a discussion as to the ways the TI-Echo was distributed to the different BBS' across the country and explained just what was going on in the battle of the SYSOPS. He explained that we are able to access the North E. states and Texas at the present, but some of the Western states are no longer on our local board.

b. Scott suggested the secretary send a post card to Mr. James Peterson who had requested a newsletter swap in March. We have sent issues May,June, and July with no response from Mr. Peterson.

c. The TI package that a friend of Tony Kleen was selling was purchased by John Gollar. Since Emmett was not present, there was no news as to the prices asked by his friend for other TI equipment for sale.

d. Bob told members that a left-hand threaded shaft for the McInker has been ordered and is on the way. That will make the unit compatable to clockwise clockwise and counter-clockwise rotating ribbon carts.

New Business:

a. The secretary passed around a flyer from Asgard Software on the special offer to User Groups for the Page Pro 99.

b. Tony Kleen agreed to present the program for August.

The APL Editor was demonstrated by yours truly and contained a short Adv. program that started at a parked car in the parking lot of the Center, and ended in the TI meeting room when the computer grabbed me and stuffed me into the disk drive sending me to Never, Never Land! The members seemed very interested in the making up of an APL program and the demo seemed to be well received.

After the demo, Bob gave some info on changes he is trying with TI-Base. I am sure Bob will do a good job and let us in on his progress.

The meeting was adjurned at 9:45 P.M.

Respectfully submitted,
L. F. "Mac" Jones, Sec./Treas.
Guilford 99er Users' Group
Greensboro, N. C.

FORTH TIPS 9

By Lutz Winkler

THE <BUILDS...DOES> CONSTRUCT IN TI-FORTH

This tutorial is being written upon request by a TI-FORTH enthusiast who says that he has a problem figuring out <BUILDS...DOES>. It has been a long time since I concocted the last "tut", but I am not quite out of practice (having sporadically engaged in issuing "tidbits") and what follows is my humble attempt to oblige.

Somebody (I wish I could remember who) once wrote that "the most powerful programming tool you have in FORTH is the ability to define new defining words." The problem lies in "separating the compile-time action of the DEFINING word from the run-time action of the DEFINED word." How true. Let's see if we can shed some light on the matter.

If you are familiar with the DEF statement in Extended Basic we can use it as a starting point because <BUILDS...DOES> is somewhat akin to it. When you define a function in XB nothing visible happens when the program is loaded. Yet the function has been placed in memory ready to perform if called upon. In other words, it has been compiled. However, as soon as the program runs and encounters the name of the function it will be executed as defined - the run-time action. This may be a bit of oversimplification but should give you an idea of what is meant by compile-time and run-time action.

<BUILDS...DOES> allows you to define defining words which is indeed a powerful tool. What it means is that the defining word, when invoked, creates a new word which does exactly what the defining word specifies. I am sure you have used defining words already without being aware of it: VARIABLE and CONSTANT are defining words. Since they are coded in assembly and part of the kernel you may not have realized that both use <BUILDS...DOES>. If they were written in high-level and listed somewhere on one of the screens, they would look like this:

```
: VARIABLE <BUILDS , DOES> ; ( n --- )

: CONSTANT <BUILDS , DOES> @ ; ( n --- )
```

As you can see, there is very little difference in their definitions at least in the compile-time action which is the part between <BUILDS and DOES>. Both store the parameter n in the first available memory cell after <BUILDS has created a new dictionary entry. Thus, no matter whether you use 5 VARIABLE FIVE or 6 CONSTANT SIX, each contains or represents a certain value. The significant difference lies in their run-time action. In VARIABLE there is no run-time action (nothing follows DOES>), while CONSTANT performs a fetch during run-time. The result is that when you invoke FIVE only its address is left on the stack and you must use a @ or ? to retrieve or retrieve and display its current value. On the other hand, SIX also leaves its address on the stack but the @ (after DOES>) uses it to fetch its value and puts it on the stack. This is precisely what you have known all along: You must fetch the value of a variable when you need it while a constant's value is put on the stack by invoking its name. Now let's see how we can separate the compile-time and run-time actions of a defining word:

```
def-word <BUILDS (compile-time action) DOES> (run-time action)
```

or - in terms of VARIABLE and CONSTANT - this is what happens:

```
5 VARIABLE FIVE <BUILDS (create a dictionary entry named FIVE,
store 5)
DOES> (no action, i.e., FIVE's address is left on
the stack and @ must be used)

6 CONSTANT SIX <BUILDS (create a ditionary entry named SIX,
store 6)
DOES> @ (fetch, i.e., 6 is left on the stack)
```

Now look at the definitions of the words 2VARIABLE and 2CONSTANT in Appendix C, page 5, of the TI-FORTH manual (Notes on Starting Forth). These words are intended for double-precision (32-bit) numbers but you will find that they follow the same pattern with two exceptions: when you initialize a 2VARIABLE it automatically defaults to zero - the 0. following <BUILDS does that - and CONSTANT uses 2@ instead of @. In order to utilize double-numbers you must, of course, use double-number operators.

Are you beginning to see the light? Take a look at the welcome screen (#3). Here we find colon definitions galore to boot the load options like -EDITOR, -GRAPH, etc. They all perform the same task, i.e., boot something beginning with a specified screen number. Let us define a defining word and do away with those definitions. We know that the words being defined must contain a screen number and at run-time will have to do a LOAD starting from that screen. Let's call our defining word BOOTS:

```
: BOOTS <BUILDS , DOES> @ LOAD ; ( n --- )
```

The compile-time action consist of storing the beginning screen number n which is done by the comma after <BUILDS. (If you don't know about comma, look up its definition in the manual's glossary.) At run-time the screen number is fetched and LOAD is performed. Now BOOTS can take care of the load options if we use statements like those below instead of the colon definitions.

```
33 BOOTS -SYNONYMS (same as : -SYNONYMS 33 LOAD ; )
34 BOOTS -EDITOR (same as : -EDITOR 34 LOAD ; )
etc. etc.
```

By now you may have gotten the misleading impression that <B..D> can only store parameters. Don't be fooled, it can do anything a colon definition is capable of as the following examples will show. Here, for instance, is a defining word to initialize two-dimensional arrays:

```
: 2D-ARRAY ( #rows #columns --- )
<BUILDS 2DUP ( dup the parameters )
, ,
* ALLOT
DOES> ;
```

Note the 2DUP word which is not part of TI-FORTH. You can substitute OVER OVER or add 2DUP (: 2DUP OVER OVER ;) to your dictionary before defining 2D-ARRAY. As you can see, the compile time action contains a little more than just a comma or two to store parameters: it computes the number of bytes to allot to accommodate the array. (If you expect to have numbers greater than 255 you would have to multiply by 2 again in order to allot space for 16-bit words.) While it is not imperative to store the dimensions of the array in the first memory cells as this word does, we'll see shortly why this is a good idea.

Assume we want to use this array to store the scores (tests or golf or whatever) of four people over a period of 26 weeks, meaning that the array needs to be 4x26, and we'll call it SCORES.

```
4 26 2D-ARRAY SCORES
```

creates the array. To make sure it contains the correct dimensions, we can check the first two memory locations with

```
SCORES ? 26 ok
```

and SCORES 2+ ? 4 ok

To digress a bit from <BUILDS...DOES> for the moment, here is the reason for placing the array's dimension in its first two cells: they can be used to access any location within the array with the following word:

```
: CELL ( row# col# --- addr )
DUP >R ( dup address and save it to the return stack )
@ ( fetch number of columns )
ROT ( bring row# to top of stack )
* + ( multiply #col by row# and add to col# )
R> ( retrieve address from return stack )
4 + ( address of first member )
+ ; ( add offset )
```

Note that this is NOT a defining word and it is set up to access bytes not 16-bit words. Now we can use SCORES and CELL to enter numbers in the array, just keep in mind that in FORTH we always start the count at zero:

```
75 0 2 SCORES CELL C! puts 75 in row 1, column 3
90 1 1 SCORES CELL C! puts 90 in row 2, column 2
```

To check if the values were placed in the correct cells, we enter:

```
0 2 SCORES CELL C@ .  75 ok
1 1 SCORES CELL C@ .  90 ok
```

With the defining word 2D-ARRAY you can set up arrays to you heart's content. But since my purpose here is to explain <BUILDS...DOES> I won't go into any further details about arrays. Instead I will give you a defining word with pratical value. Its author is Michal (sic) Jaegerman of Edmonton, Alberta, Canada.

```
: PRINTS ( n1, n2, .. number of n's ---- )
<BUILDS HERE SWAP DUP C, ALLOT
HERE 1-
DO I C! -1 +LOOP
HERE =CELLS DP !
DOES> SWCH COUNT TYPE UNSWCH ;
```

This should convince you that anything which can be done with a colon definition can also be achieved with a defining word. If you disect the compile-time portion of PRINTS you find that it stores (compiles) the number of parameters needed into the first byte, then uses that number to store the parameters with a DO-LOOP. The run-time action consists of sending the parameters to the printer. Since all printer commands are essentially alike (one or a sequence of ASCII values are transmitted to the printer) a defining word is a perfect way to create them. For example, to set my printer up for emphasized mode, I need to send ESC E. Thus

```
27 69 2 PRINTS EMPHASIZED
```

creates the word EMPHASIZED which, when invoked, sends the command to the printer. In a like manner, you can create words for all printer commands you use frequently. All that's required are the ASCII values for the command itself, the total number of values, PRINTS and a word which you select (hopefully with some mnemonic value).

I hope the foregoing has provided some help in understanding <BUILDS.. unless you want to create a number of words with similar functions. If that is not the case, stick with colon definitions.

RAMBLING BYTES

By Mac Jones

After the demo Wednesday evening, there should be quite a bit of APL - hacking going on with the 6 members who seemed to enjoy what goes on in writing one. I think it is a very interesting programming language. My Graveyard Adventure is coming along gamely. I showed the members the APL print-out of the small part of it I have finished and they were amazed at the pages of print for just a little part of the total Adventure. Thank goodness for printers!!

A gentleman has left me a message on the ROS BBS wanting to sell a bunch of TI equipment. I will list what he has at the end of my column. It seems that there is more and more of it turning up. Buddy Cato informed me not long ago that he picked up a gray console at a yard sale for $8.00! Now friends, that is a real bargain. I left a note on the Echo that I wished someone had a trash mother-board for the TI as I wanted a joystick socket off one. A user in Tulsa, Oaklahoma answered and said if I would send the postage he would be glad to send me one. I sent him 5 bucks and in a few days I received 2 boards!! One of them was complete and one had a few memory chips removed but I only wanted 1 anyhow. Anyone need a memory chip? (grin).

Well I still haven't gotten my bargain RS232 card running as yet. I talked to friend John Wilforth one day by phone and he suggested replacing the LS244's or the LS245. After removing all of those chips and inserting sockets in their place, I tried all new chips (LS244 245) but it must have gotten a lightening zap, because I understand those chips are generally the ones that cause the kind of trouble I am having. When the card is plugged in, the disk controller LED comes on and I am unable to get the TI Logo on the screen. They say all things come to those who wait, so I will just wait!!

Got a letter from some friends over seas in the EAR group, and guess what? I am now the proud owner of a "smooching license"!! Seems Joanne Copeland has the Certificate 99 program and wanted to try it out long range! Sure makes a nice certificate except you should see what I get when I try the program on my AT&T serial! First it runs off about 3 sheets of paper and then it prints about a dozen X's in a row. It skips about 5 spaces and prints out another bunch of X's. I really don't understand this printer. I never have understood why it will put out garbage after one page of printing. There have been quite a few users who were going to tell me the right way to pin it, but all the suggestions have failed. I am beginning to think it has some type of buffer trouble because it seems that when the buffer empties, there is no handshake to the computer to let it know it's empty and starts printing parts of sentances and gibberish. It is a nice printer, but I quess I am going to have to "break-bad" and get me a parallel printer that is Epson compatable for the TI.

I promised you that I would list the items that the gemtleman wanted to get rid of so I will hang it on this line and say until the next meeting, take care and enjoy the good TImes.

FOR SALE

2 BLACK AND SILVER CONSOLES 1 BIEGE CONSOLE
3 MODULES (type not specified) 1 DISK MANAGER II
2 JOYSTICKS 2 CASETTE CABLES
1 SPEECH UNIT

FOLLOWING GAME MODULES:

TOMBSTONE CITY	ADVENTURE NEW IN BOX
EARLY LEARNING	A-MAZE-ING
ALIEN ADDITION	TEACH YOURSELF BASIC
MUSIC MAKER	MISSION IMPOSSIBLE
DONKEY KONG	COMPUTER MUSIC BOX
PARSEC	NUMBER MAGIC
PERSONAL RECORD KEEPING	BLACKJACK AND POKER
FOOTBALL	

He is asking $100. for the lot. He says he wishes to sell it all in a bunch so he can get finished with it. Anyone interested can let me know and I will get you in touch with him. (Mac Jones)

NEWSLETTER DEMISE

I received the following postal card July 7th from the Charlotte Group.

DEAR USER GROUP:

The Charlotte TI Users Group has restructured due to a serious drop in membership. Accordingly we are having to cease publication of our newsletter. Therefore we must cease our exchange with your club. Our agreement was that if we send ours you will send yours. We will understand if you decide not to send your publication. However--IF you are willing to send us as a service to a diminished club--if only periodically--WE WILL GREATLY APPRECIATE IT! We have also changed our mailing address. New address is:

Charlotte TI Users
c/o Arnold Wollman
8200 Eagles Point Ct.
Charlotte, N.C. 28226

Thank you for your friendship in the past. We hope it will continue!

I am afraid we will be getting more of these notices as time goes by. The lack of interest is becoming evident as I read other newsletters. Hopefully, we can continue to meet and provide newsletters for our members and exchanging friends. Thank all of you for your support of our group. If we do go down, lets go down fighting!!

MODEMS ETC

By Tony Kleen

Confessions of a Small-TIme User. by Tony Kleen. Guilford (NC) Users Group.

We've been discussing modems, uploads, downloads, Kermit (the frog?!?), and all sorts of other 'things' at our meetings, lately. Well, I for one, do not know what all these 'things' do, or are, or whatever. So..., being diligent, and resourceful (and not wanting the members to know that I don't know what I'm talking about), I went to the library. Found "Alfred Glossbrenner's Master Guide to Free Software", which is for IBM's and compatible computers. The section I was interested in, ie., telecommunications, is applicable to all, though.

Topic - The UART and the Modem:

As you may know, our little computer batches bits together in units of eight bits each, called a BYTE. Each byte of information passes through the computer along eight wires, in parallel formation. Everything inside the computer is set up for the bits to roll along in parallel formation. That takes eight wires at least, which is seven more outgoing wires than you will find in most telephone systems. In the second place, things are pretty quiet inside the computer, electrically speaking. So each bit can be the electrical equivalent of a whisper and still be heard. In the big wide world, there is a lot of noise and resistance that would make your average computer bit inaudible.

The solution is to perform a conversion, two in fact. First, the natural parallel paths of computer bits must be changed from eight-all-at-once to one-at-a-time. This is called a conversion from parallel to serial data communications. Second, each bit has got to be changed into a more rugged form suitable for telephone transmission. Since phone systems were designed to handle sound, that's what the bits must be changed into.

There is a piece of equipment to handle each conversion. The first conversion is done by a microchip called a UART ("you-art"). This stands for "universal asynchronous receiver-transmitter." UART's are the central element of components that are variously called comm, asynch or serial cards, RS-232 interfaces, or communication ports. These chips are responsible for getting each eight-bit unit lined up, adding two bits of what for now we can think of as packing material, and sending the entire ten-bit package out the door.

Fortunately, you rarely need to worry about UART's or even be aware that they are there. What you DO need with the TI-99 is an RS-232 board, which has a built-in serial interface.

The second conversion, the one from electrical signals into sound, is performed by the modem. This term is a compression of the words MOdulator/DEModulator, and it refers to the device that changes outgoing computer signals into sound and incoming sound into computer signals. As noted, phone lines are designed for sound, not computer voltage levels. So the bits that began as voltage pulses in the computer are transformed into two, and only two, different sounds.

You can think of them as a high-pitched tone and a low-pitched tone. It is helpful to know that modems deal in two sets of paired tones or frequencies, one pair for outgoing and one for incoming information. These two sets are referred to as the modem's "originate" and "answer" modes. For communications to take place, one modem must be set to originate, and the other must be set to answer. If you are calling a distant computer, your modem should be set to "originate".

Modems are connected to the TI99/4A at the RS-232 port. It is officially called RS-232-C. The hardware is called a DB-25 connector and consists of 25 pins or sockets. A second cable leads from the modem to the telephone jack. Thus, the modem stands between the computer and the telephone system, modulating and demodulating sound and computer signals.

IF YOU WERE BORN BEFORE 1950

We were born before TV, before penicillin, polio shots, frozen foods, Xerox, plastic, contact lenses, frisbees, and the PILL.

We were before radar, credit cards, split atoms, lazer beams, ballpoint pens...before pantyhose, dishwashers, clothes dryers, electric blankets, air conditioners, drip-dry clothes and before man walked on the moon. We got married first and then lived together. How quaint can you be?

In our time, closets were for clothes, not for "coming out of," Bunnies were small rabbits and rabbits were not Volkswagons. Designer jeans were scheming girls named Jeanne or Jean, and having a "meaningful relationship" meant getting along well with our cousins.

We thought fast food was what you ate during Lent. We were before house-husbands, gay rights, computer dating, dual careers and commuter marriages. We were before day-care centers, group therapy and nursing homes. We never heard of FM radio, tape decks, electric typewriters, artificial hearts, word processors, yogurt, and guys wearing earrings. For us, time-sharing meant togetherness--not computers or condominiums. A 'chip' was a piece of wood, hardware was hardware and software wasn't even a word!

In 1940 "making out" referred to how you did on your exam...Pizzas, 'MacDonalds' and instant coffee were unheard of.

We hit the scene when there were 5 and 10 cent stores where you bought things for a nickel and a dime. Ice cream cones were 5 cents and for that same price you could ride a street car, make a phone call, buy a Pepsi or enough stamps to mail one letter and 2 postcards. You could buy a Chevy coupe for $600, but who could afford one? A pity too, because gas was 11 cents a gallon.

In our day GRASS was mowed, COKE was a cold drink, POT was something you cooked in, ROCK MUSIC was Grandma's lullaby and AIDS were helpers in the Principal's office. We were certainly not before the difference between the sexes was discovered, but we were surely before the sex change...we made do with what we had and we were the last generation that was so dumb as to think you had to have a husband to have a baby. No wonder we are so confused and there is such a generation gap today.

BUT...WE SURVIVED!!

As most of us use our computers as word processors, there are rules by which we must compose our writing. The followering rules were left by an old English Professor who did not leave his name but left us the valuable rules. Without further adeiu here are:

RULES OF GRAMMAR

1. Don't use no double negatives.
2. It's important to use apostrophe's right.
3. Watch out for irregular verbs which has cropped into our language.
4. About sentence fragments.
5. Each pronoun agrees with their antecedent.
6. Dangling, you should be careful about participles.
7. Verbs has to agree with their subjects.
8. Don't abbrev.
9. And of course, there is that old one: Never use a preposition to end a sentence with. As Winston Churchill once put it, "This is something up with which I will not put."
10. Check to see if you any words out.
11. In letters themes reports articles and stuff like that you use commas to keep a string of objects apart.
12. Join clauses good, like a conjunction should.
13. Don't use run on sentences you've got to punctuate.
14. In my opinion, I think that an author, when he is writing, should not get into the habit of making use of too many unnecessary words that he does not need to use in the article he is writing them in.
15. Last but not least, lay off cliches. (Mc)