THE GUILFORD 99'ER NEWSLETTER

VOL.4 NO.10 OCTOBER 1987

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 27408. The Software Library is for dues paying members only. (Herman Geschwind, Editor)

OUR NEXT MEETING

DATE: October 8, 1987. TIME: 7:30 PM PLACE: Glenwood Recreation Center 2010 S. Chapman Street.

When TI manufactured the consoles, someone goofed in that rather than use a 330 ohm VDP Load Resistor required by the TI specs, a 560 ohm resistor was used. Lowering the resistor value yields faster fall times, which results in a sharper picture. Our President is going to prove to us that it is not all that complicated to make the switch and as a hardware project he will install a new load resistor into the beige Club console. Time permitting he will also operate on your console for a contribution of $10 to the Club kitty.....

PRES PEEKS

There it was 7:30 PM and four people were present counting me!! After all the begging for a session on Multiplan and hoping that Herman would conduct one, where were were you??

We did have a few more to show up including a nice Lady from High Point who I hope will join us. Herman did a bang-up job on his demo and I would like to thank him again in print for his excellent program. He wasn't able to bring his ram disk or as he put it, he didn't like to gloat (none of us other members have one as yet) but it really wasn't that slow loading. However I have visited Herman and seen what a really fast job that Horizon does for Multiplan. There are two things I really want for my system and they are a Super-cart and a Ram disk. I have given up on the Gramkraker for I know I will never have one anyhow.

I would like to thank Mike Garrett for the fine job he did on last month's newsletter. He even collated and stapled them for me so it really made my job a lot easier just folding and labeling and stamping them. I get a charge out of my mail carrier each month when he sees the wad of newsletters for him to pick up!

October will soon be here and hopefully most members will be over all vacations and we can have a good turn-out again. I realize how easy it is to say "well, I will just miss this one and go next month". First thing you know, it's the same thing next month. We miss you when you are away and I feel that someone can get something out of each meeting, even if it's just a feeling of belonging and fellowship with other members.

At a meeting a couple months back, the members were asked to raise their hands if they had a modem. I remember quite a few hands in the air. The reason I say this is that there is four BBS operating in Greensboro, and there may be more that I am not familiar with. Now it doesn't cost you one red cent to use these boards and they are an excellent way to gain information and also programs. What I don't understand is only Herman and myself use them. Once in a while George McCormick will upload a program and leave a message, but none of the members are using them. OPUS has alloted us a TI space. It is guarded from just anyone using it and you may get a password from Herman to access the TI files. ROS-GROUNDSTAR has given us space also and it is running again since Dan Post installed new equipment. It was out of service for a few weeks but is now active. Also, there is a board run by a few Teens which has given us a space for TI, however, no one has used it but me so far. The other board is BILLBOARD that has programs for PC's but at present no TI section. Using a modem with the up to date terminal emulator programs that we have is no problem. I would like to see more people using them. Remember, the price is right....nil!

The current numbers for the boards are: OPUS - (919) 274-5760 ROS - (919) 261-2623 BILLBOARD - (919) 271-8118 and the one by the Teens, I believe is called Computer World - (919) 299-7935. All of these boards operate 24 hours a day and use from 300 baud up to 2400 baud. Opus has a "For sale" file that reaches most towns in North Carolina and gives you a chance to pick up some good buys from users who are 'stepping up' in the computer world. Try it out sometime.

The door prize last month was won by Larry Spohn. He received the Club's issue of MICROpendium. Mr. Benson from Eden gave us the labels he had received from TI, but the club decided not to use them. Larry Spohn asked to take them to the Winston Group which was agreed on. There were none from Greensboro included .

Well, as I said earlier, I was wanting a Horizon Ramdisk, but I am glad I waited on getting one. This month, DataBiotics came out with what I think is a better deal, costwise, than the Horizon. It's called GRAND RAM. It fits my situation more because it has expansion capabilities, that is...you can buy the initial 64k card for $129.95 or if you are so inclined..a do-it-yourself version for $99.95. Then as your money comes in, you can add memory to it until you reach the 512k complete board. You may also buy a clock chip for $19.95 (if ordered at time of board) which will afford you a means of having the time displayed while you use your console. The part that is nice about this unit is you may turn the DS/DD card into 4 single sided or 2 double sided drives in a matter of minutes. If your rich uncle left you some money, you can add cards in your P-Box up to a grand total of 2 megabytes!! It is compatable with TI, CorComp, and Myarc controller cards.

All you need to start is an expansion box and some kind of disk controller card. You will be provided a disk manager and 4A/Talk TE program. Also with the card, a fully documented source code for the device, print spooler and loader/configurator, and complete instructions. The prices run......

Pre-Built	Build your Own
64k............$129.95	64k..........$ 99.95
128K...........$144.95	128k.........$112.95
256k...........$169.95	256k.........$136.95
512K...........$229.95	512k.........$185.95
Clock Chip.......$19.95 when ordered with unit.	

Why all the .95 cents I can't understand. I had much rather see something advertised for $20. than 19.95. I guess sometimes it just looks cheaper when it's 199.95 than if it says $200.00. Anyhow, it's there for anyone who needs a Ramdisk cheap with add-on capability.

I had seen the new EPYX 500XJ Joystick advertised and I was impressed by the mention that it had momentary contact switches inside rather than the spring type on most sticks. I mentioned that it might be a good thing to have and since my birthday was coming up (July), it would make a nice gift. Well one of my Daughters presented me with one and I have been using it for a couple months now. I really don't like it on some games. It is rather sluggish if you need real fast movement such as Donky Kong and some other arcade type games. I have gone back to my Prostick II. I know Bob likes Prostick, for he did an article for the August MICROpendium on it. The only trouble I have had with the Prostick is the cord broke internally right at the stick, but I opened it up and re-soldered the leads after skinning past the broken part and it works fine. Bob, however, sent his back to the factory and for $5. they refurbished his. I could have sent mine back, but I just like to piddle with things and I also hate waiting for things to come back that have been sent away.

I got a nice letter from Mrs. JoAnn Copeland of the EAR 99'ers, the group of users overseas that we agreed on including in our mailings. I had asked if they were receiving their newsletters, and she wrote to say they were. She and her Husband are on a tour of duty in London and have quite a few 99'ers over there. It was nice to hear from them.

For you puzzle solvers out there, I ran upon this puzzle that I found stuck back in some papers I was going through. See how far you can go with it. I know a friend out in Texas name of Steve Venable will get it right away, as well as quite a few of his members in PUG. Here 'tis....

In these remarks are hidden the names of 15 books of the Bible. It's a real lulu. Kept me looking so hard for facts, I missed the revelations. I was in a real jam especially since the names were not capitalized. The truth will come to numbers of our readers. To others it will be a real job! For all it will be a most fascinating search. Yes, there will be some easy to spot. Others, hard to judge, so we admit it usually results in low lamentations when we can't find them. One lady says she brews coffee while she puzzles over it.

There you have it! I hope you can get them all. I wish I could give the credit where it is due for this puzzle, but it was written on the back of an envelope that had a 1967 postmark, so who ever did this ...thanks.

Got a message on the board from 'ol Herm last night wanting this right away, so I will hang my close on this line and upload it to him. Remember......Yesterday is a cancelled check, Tomorrow is a promissary note, and today is cash...spend it wisely.

(Submitted by "Mack" Jones)

WRITERTIPS

HOW NOT TO WASTE AN ENTIRE AFTERNOON UNDER THE GUN!

Even after years of word processing and healthy computer skepticism, it can happen. ZAP! You let your guard down, press a key innocently enough and instantly an afternoon's worth of work is vaporized. Despite all the ingrained protective measures and software failsafe defaults, it can happen.

It did to me this month, while working against--of course--a deadline. Allow me to share it with you in the interest of rekindling that on-guard healthy skepticism when dealing with computers. And let me begin by saying that short of a power surge or some unlikely electrical malfunction in your console, the guilty party is usually human. Be skeptical of yourself, because unlike the computer our ability to make dumb, costly errors is incredible.

I made mine by getting too comfortable with TI-Writer. I broke a couple of my own rules and paid the price of re-writing an eight-page piece from scratch. Here's how.

First, I loaded into TI-Writer a form file of printer formatting commands I use when writing a lengthy article. Nothing wrong with that. It saves time. In fact, I wanted to save even more time. So instead of modifying the file to suit my specific purpose, I changed the file name and partially saved it back to disk using TI-Writer's handy partial file saving ability. It allows you to save a specific block of copy by listing the line numbers prior to the file name. For example: "1 15 DSK2.MYFILE"

The only problem is that this now becomes the default file name, including the preceding copy line numbers, unless you eliminate them the next time you save the file to disk. You guessed it. I didn't. I forget to remove the line numbers. So for an afternoon, carefully following my own rule to religiously save the entire buffer file to the disk every 15 to 40 lines, I was in fact only saving lines 1 through 15. Though I was constantly adding to my buffer file (creating the article), none of it but the first 15 lines ever made it to the disk. It never occurred to me that the drive wasn't spinning long enough to be writing the long additions I was making. Indeed that humming drive just lulled me into a false sense of security. I was humming. It was humming. It was great afternoon and I was getting a lot accomplished. Hum. Hum.

The second big mistake I made was failing to do a disk directory read, upon completion of the piece. In fact, since I use two drives writing to both as an added safeguard, I normally do a directory of both disks comparing the sectors occupied by the piece on each. If they match, I know I have the piece on two different disks. But humming as I was, I didn't even do a directory of one disk. Had I, I would have discovered that an article which should have occupied many sectors was only on a few.

The third mistake I made was not making a hard copy version using the (again) convenient TI-Writer feature to print the original file from the editor utilizing the PrintFile command. Such a file includes all the formatting commands, but they represent a very minor nuisance when the alternative is rewriting from memory more than eight original pages of text, or more.

A fourth mistake was leaving TI-Writer without making a final save to disk via the quit command function. It prompts the user to select one of three options: Save, Purge or Quit. It isn't just fancy that the TI-Writer programmers put the save function first in this lineup and had I availed myself of the option, I might have noticed in this different environment that all afternoon I had been saving a defaulted partial file. But I was hummmmmmming.

Imagine my shock when I jump right to the formatter, tapped a few keys and got up to get a much deserved beer only to hear the printer come to a crescendo of silence. "Huh?" I muttered. "Printer dead," I wondered. "RS232 shot so soon," I speculated. "Printer print," I commanded. "Dumb machine," I uttered attaching a stream of assorted four-letter adjectives.

Back into the editor I raced to do a belated directory of the disk, discovering that a whimp of a file was where there should have been a mighty, many-sectored masterpiece. Sweating, I made the rounds of TI user groups friends who confirmed what I already knew: once you leave the editor, you done kissed it goodbye; or, if it ain't on the disk, it ain't!

So TI-Writer writers, don't get complacent. Computers, I have found, love nothing better than to teach smug hummers a lesson. Follow these rules:

1. If you use form files or partially save a file to disk, make sure on subsequent saves that it is getting filed completely, the way you want it.

2. Even if you do not have a multi-drive system, still write those lengthy pieces at least to two dependable disks.

3. Before leaving the editor perform a disk directory to ensure the file is there, it is about the length it ought to be and occupies the exact same number of sectors on both disks.

4. Also before leaving the editor, consider sending a rough draft to the printer as insurance. Remember no system is failsafe. Something could be wrong with both disks and hard copy generally is a lot better than your memory.

5. Before Quitting , save the document one last time making sure it is being filed the way you intend it to be.

6. Really enjoy that beer. (Contributed by Larry Spohn)

FORTH FORUM

This month, I thought that we would take a look at some of the truly outstanding TI-Forth programming from "down under". Dr. Richard Terry has been writing a Forth column for the HV99'ers UG newsletter for quit awhile now and the following screens appeared in his column some time ago. This routine provides the facility to draw boxes around titles, etc. in your programs for a more professional appearance. It is extremely easy to use. To create a box all you have to do is enter the corners followed by DBOX. It would look something like this: 5 5 24 19 DBOX.

Screen #14

```
0 ( Box Drawing routine 14MAR86 R.Terry ) HEX
1 FF00 FF00 0000 0000 84 CHAR ( 132 Top Line )
2 FF80 BFA0 A0A0 A0A0 7B CHAR ( 123 Upper Right Corner )
3 A0A0 A0A0 A0BF 80FF 7C CHAR ( 124 Bottom Left Corner )
4 FF04 F414 1414 1414 7D CHAR ( 125 Upper Left Corner )
5 1414 1414 14F4 04FF 7E CHAR ( 126 Bottom Right Corner )
6 0000 0000 00FF 00FF 80 CHAR ( 128 Base Line )
7 A0A0 A0A0 A0A0 A0A0 81 CHAR ( 129 Left Vertical )
8 1414 1414 1414 1414 82 CHAR ( 130 Right Vertical )
9 00FF 0000 0000 0000 83 CHAR ( 131 Underline )
10 8080 8080 8080 8080 85 CHAR ( 133 Single Line Right Vertical )
11 0404 0404 0404 0404 86 CHAR ( 134 Single Line Left Vertical )
```

```
12 0000 0000 0000 00FF 87 CHAR ( 135 Single Line Baseline )
13 8080 8080 8080 80FF 88 CHAR ( 136 Single Line Lower Left Angle)
14 0404 0404 0404 04FF 89 CHAR ( 137 Single Right Left Angle )
15 8000 8000 8000 8000 92 CHAR ( 146 Dot Vertical ) DECIMAL -->
```

Screen #15

```
0 ( Box Drawing Routine 14 MAR86 R.  Terry )
1 : P 2 * SP@ + @ ; ( PICK word redefined )
2 : DBOX ( Left column, upper row, right column, lower row )
3 4 P 4 P 1 123 HCHAR ( upper left char )
4 4 P OVER 1 124 HCHAR ( lower left char )
5 OVER 4 P 1 125 HCHAR ( upper right char )
6 OVER OVER 1 126 HCHAR ( lower right char )
7 4 P 1+ OVER 4 P 1- 7 P - 128 HCHAR ( lower horizontal )
8 4 P 4 P 1+ 3 P 1- 6 P - 129 VCHAR ( right vertical )
9 OVER 4 P 1+ 3 P 1- 6 P - 130 VCHAR ( left vertical )
10 4 P 1+ 4 P 4 P 1- 7 P - 132 HCHAR ( upper horizontal )
11 DROP DROP DROP DROP ;
12
13
14
15
```

The only pre-requisite for these screens is that -GRAPH must be loaded for this application to work. Next month, we will take a look at some more TI-Forth by taking DBOX one step farther (with Dr. Terry's help). (Contributed by Bob Carmany)

BASIC/XB CORNER

(Editor's Note: For space reason I could not comment on our program listing last month. Well, it was a program that TI distributed to their dealers way back when the speech module was first introduced. In a rather amusing way it shows off the capability of XB with the speech synthesizer attached. You might want to take a real close look as to what is possible using no more than the built-in vocabulary..

We are indeed real fortunate that Bob Carmany was able to secure the McGovern Extended Basic tutorials for us. Originally published in Australia, these tutorials have been published in many newsletters and magazines in Europe, the US and Canada and are generally acclaimed to be the best. They are not a beginner's introduction to XB, rather they serve to supplement what TI did not include in the manual or tricks and tips that have not been covered in any other publication. I have left the source material intact and only edited the text to make it fit the space available. While some of the references are to conditions Australian and the spelling and phrasing might be "down under", I elected to leave it in the original to convey the full flavor of T. McGovern's unique style.

In order to bring as much of the tutorial material as possible, we will not publish a program listing in this and the next issue. Enjoy.)

EXTENDED BASIC TUTORIALS

I. INTRODUCTION

In this series of notes on TI Extended Basic for the TI-99/4a we will concentrate on those features which have not received due attention in User-group newsletters or commercial magazines. In fact most of the programs published in these sources make little use of that most powerful feature of XB, the user defined sub-program, or of some other features of XB. Worse still is to find commercially available game programs which are object lessons in how to write tangled and obscure code.

The trigger for this set of tutorial notes was a totally erroneous comment in the TI.S.H.U.G Newsdigest in Jun 1983. Some of the books I have seen on TI Basic don't even treat that simpler language correctly, and I don't know of any systematic attempts to explore the workings of XB. The best helper is TI's Extended Basic Tutorial tape or disk. The programs in this collection are unprotected and so open for inspection and it's worth looking at their listings to see an example of how sub-programs can give an easily understood overall structure to a program.

Well, what are we going to talk about then ? Intentions at the moment are to look at

(1) User-defined sub-programs
(2) Prescan switch commands
(3) Coding for faster running
(4) Bugs in Extended Basic
(5) Crunching program length
(6) XB and the Peripheral Box
(7) Linking in Assembler routines

Initially the discussion will be restricted to things which can be done with the console and XB only. Actually, for most game programming the presence of the memory expansion doesn't speed up XB all that much as speed still seems to be limited by the built-in sub-programs (CALL COINC, etc) which are executed from GROM through the GPL interpreter. The real virtue of the expansion system for game programming, apart from allowing longer programs, is that GPL can be shoved aside for machine code routines in the speed critical parts of the game, which are usually only a very small part of the code for a game. Even so careful attention to XB programming can often provide the necessary speed. As an example, the speed of the puck in TEX-BOUNCE is a factor of 10 faster in the finally released version than it was in the first pass at coding the game.

Other topics will depend mainly on suggestions from the people following this tutorial series. Otherwise it will be whatever catches our fancy here at Funnelweb Farm.

II. SUB-PROGRAMS in OVERVIEW

Every dialect of Basic, TI Extended Basic being no exception, allows the use of subroutines. Each of these is a section of code with the end marked by a RETURN statement, which is entered by a GOSUB statement elsewhere in the program. When RETURN is reached control passes back to the statement following the GOSUB. Look at the code segments

```
290 ....
300 GOSUB 2000
310 ....
2000 CALL KEY(0,X,Y):: IF Y=1 THEN RETURN ELSE 2000
```

This simple example waits for and returns the ASCII code for a fresh keystroke, and might be called from a number places in the program. Very useful, but there are problems. If the line number of the subroutine is changed, other than by RESequencing of the whole program (and many dialects of Basic for microcomputers aren't even that helpful) then the GOSUBs will go astray. Another trouble, which you usually find when you resume work on a program after a lapse of time, is that the statement GOSUB 2000 doesn't carry the slightest clue as to what is at 2000 unless you go and look there or use REM statements. Even more confusingly the 2000 will usually change on RESequencing, hiding even that aid to memory. There is an even more subtle problem -- you don't really care what the variable "Y" in the subroutine was called as it was only a passing detail in the subroutine. However, if "Y" is used as a variable anywhere else in the program its value will be affected. The internal workings of the subroutine are not separated from the rest of the program, but XB does provide four ways of isolating parts of a program.

(1) Built-in sub-programs
(2) DEF of functions
(3) CALL LINK to machine code routines
(4) User defined BASIC sub-programs

The first of these, built-in sub-programs, are already well known from console Basic. The important thing is that they have recognizable names in CALL statements, and that information passes to and from the sub-programs through a well defined list of parameters and return variables. No obscure Peeks and Pokes are needed. The price paid for the power and expressiveness of TI Basic and XB is the slowness of the GROM/GPL implementation.

DEF function is a primitive form of user defined sub-program found in almost all BASICs. Often its use is restricted to a special set of variable names, FNA,FNB,... but TI Basic allows complete freedom in naming DEFed functions (as long as they don't clash with variable names). The "dummy" variable "X" is used as in a mathematical function, not as an array index

```
100 DEF CUBE(X)=X*X*X
```

doesn't clash with or affect a variable of the same name "X" elsewhere in the program. "CUBE" can't then be a variable whose value is assigned any other way, but "X" may be. Though DEF does help program clarity it executes very slowly in TI Basic, and more slowly than user defined sub-program CALLs in XB.

CALL LINK to machine code routines goes under various names in other dialects of Basic if it is provided (eg USR( ) in some). It is only available in XB when the memory expansion is attached, as the TI-99/4a console has only 256 bytes of CPU RAM for the TMS9900 lurking in there. We will take up this topic later.

You should have your TI Extended Basic Manual handy and look through the section on SUB-programs. The discussion given is essentially correct but far too brief, and leaves too many things unsaid. From experiment and experience I have found that things work just the way one would reasonably expect them to do (this is not always so in other parts of XB). The main thing is to get into the right frame of mind for your expectations. This process is helped by figuring out, in general terms at least, just how the computer does what it does. Unfortunately most TI-99/4a manuals avoid explanations in depth presumably in the spirit of "Home Computing". TI's approach can fall short of the mark, so we are now going to try to do what TI chickened out of.

The user defined sub-program feature of XB allows you to write your own sub-programs in Basic which may be CALLed up from the main program by name in the same way that the built-in ones are. Unlike the routines accessed by GOSUBs the internal workings of a sub-program do not affect the main program except as allowed by the parameter list attached to the sub-program CALL. Unlike the built-in sub-programs which pass information in only one direction, either in or out for each parameter in the list, a user sub-program may use any one variable in the list to pass information in either direction. These sub-programs provide the programming concept known as "procedures" in other computer languages, for instance Pascal, Logo, Fortran. The lack of proper "procedures" has always been the major limitations of BASIC as a computer language. TI XB is one of the BASICs that does provide this facility. Not all BASICs, even those of very recent vintage are so civilised. For example the magazine Australian Personal Computer in a recent issue (Mar 84) carried a review of the IBM PCjr computer just released in the US of A. The Cartridge Basic for this machine apparently does not support procedures. Perhaps IBM don't really want or expect anyone to program their own machine seriously in Basic. You will find that with true sub-programs available, that you can't even conceive any more of how one could bear writing substantial programs without them (even within the 14 Kbyte limit of the unexpanded TI-99/4a let alone on a machine with more memory).

The details of how procedures or sub-programs work vary from one language to another. The common feature is that the variables within a procedure are localised within that procedure. How they communicate with the rest of the program, and what happens to them when the sub-program has run its course varies from language to language. XB goes its own well defined way, but is not at all flexible in how it does it.

Now let's look at how Extended Basic handles sub-programs. The RUNning of any XB program goes in two steps. The first is the prescan, that interval of time after you type RUN and press ENTER, and before anything happens. During this time the XB interpreter scans through the program, checking a few things for correctness that it couldn't possibly check as the lines were entered one by one, such as there being a NEXT for each FOR. The TI BASICs do only the most rudimentary syntax checking as each line is entered, and leave detailed checking until each line is executed. This is not the best way to do things but we are stuck with it and it does have one use. At the same time XB extracts the names of all variables, sets aside space for them, and sets up the procedure by which it associates variable names with storage locations during the running of a program. Just how XB does this is not immediately clear, but it must involve a search through the variable names every time one is encountered, and appears to trade off speed for economy of storage.

XB also recognizes which built-in sub-programs are actually CALLed. How can it tell the difference between a sub-program name and a variable name? That's easy since built-in sub-program names are always preceded by CALL. This is why sub-program names are not reserved words and can also be used as variable names. This process means that the slow search through the GROM library tables is only done at pre-scan, and Basic then has its own list for each program of where to go in GROM for the GPL routine without having to conduct the GROM search every time it encounters a sub-program name while executing a program. In

Command Mode the computer has no way provided to find user defined sub- program names in an XB program in memory even in BREAK status. XB also establishes the process for looking up the DATA and IMAGE statements in the program.

Well then, what does XB do with user sub-programs? First of all XB locates the sub-program names that aren't built into the language. It can do this by finding each name after a CALL or SUB statement, and then looking it up in the GROM library index of built-in sub-program names. You can run a quick check on this process by entering the one line program

```
100 CALL NOTHING
```

TI Basic will go out of its tiny 26K brain and halt execution with a BAD NAME IN 100 error message, while XB, being somewhat smarter, will try to execute line 100, but halts with a SUBPROGRAM NOT FOUND IN 100 message.

The XB manual insists that all sub-program code comes at the end of the program, with nothing but sub-programs after the first SUB statement (apart from REMarks which are ignored anyway). XB then scans and establishes new variable storage areas, starting with the variable names in the SUB xxx(parameter list), for each sub-program from SUB to SUBEND, as if it were a separate program. It seems that XB keeps only a single master list for sub-program names no matter where found, and consulted whenever the interpreter encounters a CALL during program execution. Any DATA statements are also thrown into the common data pool. Try the following little program to convince yourself.

```
100 DATA 1
110 READ X :: PRINT X :: READ X :: PRINT X
120 SUB NOTHING
130 DATA 2
140 SUBEND
```

When you RUN this program it makes no difference that the second data item is apparently located in a sub-program. IMAGEs behave likewise. On the other hand DEFed functions, if you care to use them, are strictly confined to the particular part of the program in which they are defined, be it main or sub. During the pre-scan DEFed names are kept within the allocation process separately for each subprogram or the main program. Once again try a little programming experiment to illustrate the point.

```
100 DEF X=1 :: PRINT X;Y :: CALL SP(Y) :: PRINT X;Y
110 SUB SP(Z) :: DEF X=2 :: Z=X :: DEF Y=3
120 SUBEND
```

This point is not explicitly made in the XB manual and has been the subject of misleading or incorrect comment in magazines and newsletters. A little reflection on how XB handles the details will usually clear up difficulties.

TI BASICs assign nominal values to all variables mentioned in the program as part of the prescan, zero for numeric and null for strings, unlike some languages (some Basics even) which will issue an error message if an unassigned variable is presumed upon. This means that XB can't work like TI LOGO which has a rule that if it finds an undefined variable within a procedure it checks the chain of CALLing procedures until it finds a value. However, unlike Pascal which erases all the information left within a procedure when it is finished with it, XB retains from CALL to CALL the values of variables entirely contained in the sub-program. The values of variables transferred into the sub-program through the SUB parameter list will of course take on their newly passed values each time the sub-program is CALLed. A little program will show the difference.

```
100 FOR I=1 TO 9 :: CALL SBPR(0):: NEXT I
110 SUB SBPR(A):: A=A+1 :: B=B+1 :: PRINT A;B
120 SUBEND
```

The first variable printed is reset to 0 each time SBPR is called, while the second, B, is incremented from its previous value each time. Array variables are stored as a whole in one place in a program, within the main program or sub-program in which the DIMension statement for the array occurs. XB doesn't tolerate attempts to re-dimension arrays, so information on arrays can only be passed down the chain of sub-programs in one direction. Any attempt by a XB sub-program to CALL itself, either directly or indirectly from any sub-program CALLed from the first, no matter how many times removed, will result in an error. Recursive procedures, an essential part of TI LOGO, are NOT possible with XB sub-programs , since CALLing a sub-program does not set up a new private library of values.

All of this discussion of the behaviour of TI Extended Basic comes from programming experience with Version 110 of XB on a TI-99/4a with 1981 title screen. Earlier Versions and consoles are not common in Australia, but TI generally seems to take a lot of trouble to keep new versions of programs compatible with the old. On the other hand TI has also been very reticent about the details of how XB works. The Editor/Assembler manual has very little to say about it, less by far even than it tells about console Basic. I am not presently aware of any discussion of the syntax of the Graphics Programming Language (GPL), let alone of the source code for the GPL interpreter which resides in the console ROM of every 99/4a.

Another simple programming experiment will demonstrate what we mean by saying that XB sets up a separate Basic program for each sub-program. RUN the following

```
100 X=1 :: CALL SBPR :: BREAK
110 SUB SBPR :: X=2 :: BREAK :: SUBEND
```

When the program BREAKs examine the value of variable X by entering the command PRINT X, and then CONtinue to the next program BREAK, which this time will be in the main program, where you can once again examine variable values.

We will now summarize the properties of XB sub-programs as procedures in complete XB programs, leaving the details of joining up the various procedures to the next section.

(a) XB treats each sub-program as a separate program, building a distinct table of named (REFed) and DEFed variables for each.

(b) All DATA statements are treated as being in a common pool equally accessible from all sub-programs or the main program as are also IMAGE statements, CHARacters, SPRITEs, COLORs, and File specifications.

(c) All other information is passed from the CALLing main or sub-program by the parameter lists in CALL and SUB statements. XB does not provide for declaration of common variables available on a global basis to all sub-programs as can be done in some languages.

(d) Variable values confined within a sub-program are static, and preserved for the next time the sub-program is CALLed. Some languages such as Pascal delete all traces of a procedure after it has been used.

(e) XB sub-programs may not CALL themselves directly or indirectly in a closed chain. Subject to this restriction a sub-program may be CALLed from any other sub-program.

(f) The MERGE command available in XB with a disk system (32K memory expansion optional) allows a library of XB sub-programs to be stored on disk and incorporated as needed in other programs.

(TO BE CONTINUED)