Mid-South 99 Users Group
P. O. Box 38522
Germantown, Tn. 38183-8522

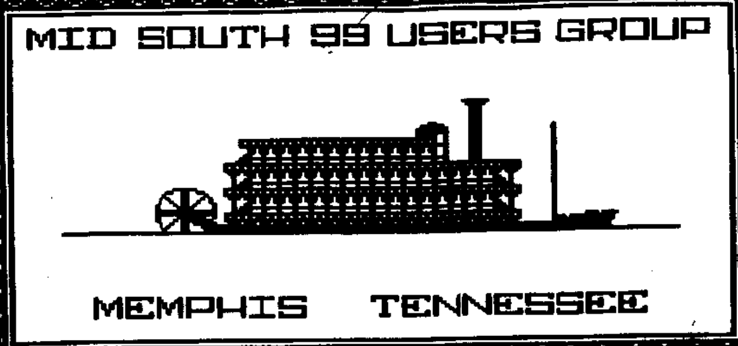
FIRST CLASS MAIL

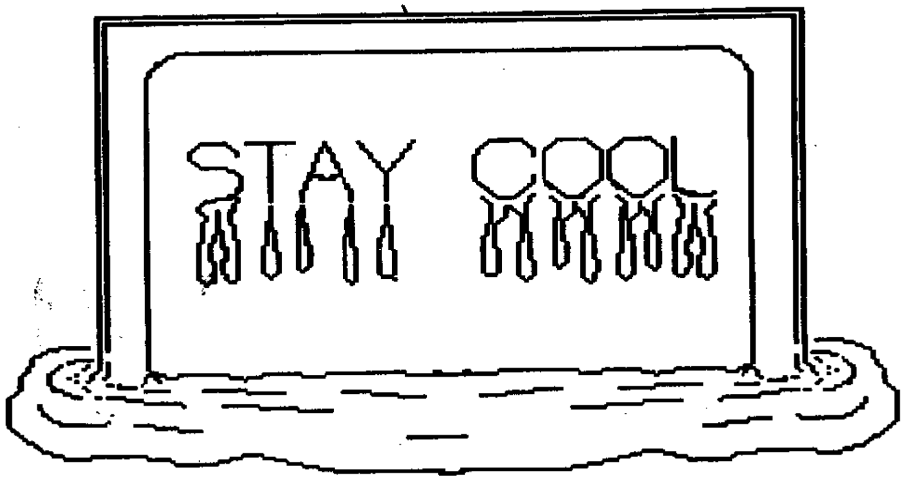
MEMPHIS
1993
AUG 16
PM
381

STRIKE BACK
AFRICAN

UG 2/86
DALLAS TI USER GROUP
P.O. BOX 29863
DALLAS, TX 75229

TIDBITS





AUGUST 1993

Newsletter of the MID-SOUTH 99 USERS GROUP-Vol 11, #8-AUG 1993

TIDBITS

OFFICERS

Gary Cox	PRESIDENT	901-358-0667
Richard Hiller	VICE-PRESIDENT	901-794-9945
Richard Mann	SECRETARY	901-682-4195
Mac Swope	TREASURER	901-363-3880
Jim Saemenes	Technical Support	901-476-7011
Jim Saemenes	Disk Librarian	901-476-7011
Pierre Lamontagne	CO-Librarian	901-386-1513
Gary Cox	Program Chairman	901-358-0667
Mac Swope	Chairman - Equipment	901-363-3880
Marshal Ellis	Editor - TIDBITS Newsletter	901-327-2506
Marshal Ellis	Editor-Technical Interface	901-327-2506
Beery Miller	9640 NEWS BBS Sysop	901-368-0112

AUG. 1993 INDEX

BACK TO BASICS

-- Art Byers

reprinted in the ERIE 99'er U.G. newsletter, Apr, 1993

TI Extended BASIC / Microsoft GWBasic : A DISCUSSION

Last year I wrote a four part series extolling the virtues of the Basic language and of TI Extended in particular for use on the 99/4A. Those pieces have been reprinted in many TI user group newsletters, so most 99'ers are well aware that Basic is one of my favorite computer languages.

It is only natural, therefore that I would investigate Microsoft GW Basic which is most often supplied as a "freebee" when an IBM compatible PC is purchased. In the small space of a computer club newsletter, I cannot go into depth of detail. After all, the GW Basic manual I have is over 240 pages. My intent is to give you my own views and let you go further from there, it you so desire.

I have heard many conflicting stories about the origination of Basic for the TI. Here is what I believe is most likely true: Regular TI-Basic, or Console Basic as it is often called, was written by Microsoft for Texas Instruments. TI Extended Basic, which is obviously based on TI-Basic, was written by a team of TI in-house programmers. It is no suprise, therefore, that GWBasic and TI-XBasic are very much alike. They are so similar that many of the more simple programs can run of bloth TI and PC systems with only minor syntax modifications.

Here are two examples: (1) XB uses a double colon "::" as a seperator for multi statements on the same line. GWB uses a single colon ":". (2) GWB uses a semi colon after text in an INPUT where XB uses a colon. ie: in GWB 100 INPUT "Enter your name"; N$ - in XB 100 "Enter your name": N$.

The overwhelming majority of statements, commands and functions, are either identical or there is a simple substitution of name such as SEG$ in TIXB and MID$ in GWB. Both versions can link assembly routines. Both can avail the programmer of useful peeks and pokes into varied memory locations. Both have the ease of program debugging generic to interpreted languages. All this is welcome news to a Basic devotee interested in transporting programs from one make of computer to another.

In fact, when reading the rest of this article, keep in mind that MOST of XB and GWB statements, commands and functions are the same right down to the lost parenthesis and comma!!!

WHICH IS MORE POWERFUL?

Although there are areas where I consider TI-XB to be definitely much superior to GWB, there can be no argument or question that overall GWB has more features, statements, commands and available memory that TI-XB. (As an aside I'll say that the new Advanced Basic for the MYARC 9640 is better than and more powerful that GWB. However, most of us are using the 99/4A and must use XBasic.) Nevertheless, if you suppliment TI XB with some asseembly graphics routines Extended Basic not inly can hold its own very well, but also in two very important respects outstrips GWB. Let's take a quick look.

AMOUNT OF MEMORY AVAILABLE.

First of all, sheer memory space available for Basic programing and required array storage is 60k in GWB as compared to about 37k in XB. Of the 48k memory in the 99/4A, Extended Basic uses either 768 bytes of 960 bytes of VDP RAM just for screen display (32 or 40 column mode). VDP RAM is also used for string variable array storage, DSR buffers, line number tables, and color and screen tables, etc. Effectively, if you have opened a line to a printer and opened a disk files (at 512 bytes each) you can safely figure at least 13k available for string storage in VDP RAM. XB uses 24k high memory for programming as well as storage of numerical arrays and constants. The lower 8k is there for assembly suport lined to basic programs but not for Basic programming. Even if we make the usual assumption that XB has the full 48k of RAM available one way or another for yse by XB programming, GWB has 25% more. You may laugh that a 640k RAM PC has so much overhead that only 10 percent is available for Basic programming. That is not really the point. GWB simply has more memory available, period! (Actually the TI has more than 90k of total memory with various ROM chips, but let's not get into that right now as no way does it have 640k of cpu addressable RAM).

The above is much mitigated by the fact that that the 99 was designed as a HOME computer. Very few programs written for home use should require 60k of memory. In six years of programming, I have NEVER run out of memory on the 99/4A for an XB program - though at times I had to do memory squeezing tricks and be very efficient with my code. If the amount of digging you have to do will be handled by a garden spade, is not a disel powered back hoe overkill?

WHERE GWB SHINES

Many, many more features make it an easy and convenient progrmming language. Let me give you just a few examples:

(1) GWB simply has more goodies for the programmer such as statments and functions to use the PC's built in clock including retireval of time and date, elapsed time, as well as the ability to set the clock. Another useful statment is SWAP, which facillitates sorting data. There are many more examples I could list here.

(2) A GWB program can use Shell commands to temporarily leave the Basic program and use a batch file, or use the many MSDOS commands such as reading directories, rename files or disks, etc and return to the execution of the Basic program when the outside task has been completed.

(3) The 99/4A has excellent file handling. So does GWB but XB lacks GWB's ability to LOCK and UNLOCK files. Even woefully deficient Apple II Basic has that feature. Why Microsoft left it out of TI Basic, I do not know. Another omission from XB is the RESET statement to close all open files and update the disk directory. That is an important adjunct to any error trap routine or any pegram that requires swapping of disks.

(4) A whole host of graphic commands such as CIRCLE, DRAW, PAINT etc are part of GWB. To use them in TI XB we must link to assembly routines such as Quality Soft's Draw and Plot or Mechatronics Expanded Graphics which contains many links to

Basic. The latter uses so much overhead, however, that the memory remaining for XB programming is severely restricted. A fine alternative is the Super Extended Basic Module which automatically dumps graphic routines into lo-memory and has other enhancements.

(5) Another real plus for GWB is the ability to use a few simple Basic statements and commands to save a whole screen onto disk, to recall the screen, text or graphics make no difference.

(6) There are many more features made possible because GWB is a larger version of Basic. These include additional looping commands such as WHOLE / WEND. All I can advise is that you take a look at a manual in a bookstore if you want to investigate further.

(7) What may well be GWB's strongest point is that it is tied to a very powerful MS-DOS. It is this DOS connection that may be, for a majority of Basic programmers, the deciding factor in favor of GWB.

(8) GWB is substantially faster in execution than TI XB. I have written articles before in "Apparant Speed". I won't repeat them here more than to say I consider TI XB sufficiently fast for almost every home use except Telecommunications. A telecom program can be written in GWB that will handle up to 2400 baud as you must program in stop/start (Ctrl S and Ctrl Q) transmissins to the remote to allow GWB to catch up with itself at 2400 baud). Above that a compiled language program is required. XB could handle 110 baud or 300 baud.

A LIMITATION OF GWBASIC

Much to my suprise, GWB limits you to a maximum of 255 dimensions in an array, (DIM A$ (255)), this applies to both string and numeric variables. To those of us in XBasic used to making DIMension statements much larger than this, either single (DIM A$ (400)) or multi dimensional (DIM A$ (100,4)) this is a potentially disabling choke collar on a very fast greyhound.

I am inclined to believe this is deliberate. If the TI with its limited RAM can dimension huge arrays, certainly Microsoft could have done the same with anywhere from 256k to 640k in the standard XT clone. Why?? Perhaps because they contemptiously don't think of the home user as writing sophisticated programs that would require large arrays. Perhaps they want to be sure that for sophisticated programs, users must go out and BUY software written in other languages, for its is a truism that the average user will program in Basic, IF they program at all.

There are ways of programming around this restriction but they sure make things unnecessarily complicated and difficult.

WHERE XB SHINES

TI-XB has several areas where even after all these years, it still outstrips GWB.:

(1) The 99/4A, although a design of the 70's, was far ahead of its time in graphics and graphic definition. Even without the high number of pixels that the latest PC's have on their monitor screens, the TI can hold its own. What is more, the 99/4A's use of and ability to control Sprites is the equal or superior , even today, of many new computers. The normal IBM clone does not yet have sprites!!! What is more, all this graphic and sprite capability (such as the MAGNIFY command) is

part of XB. GWB does not have sprites.

(2) XB statements and commands for screen display and input are superior on the TI. I consider TI's DISPLAY AT(row, col) SIZE() very much better than GWB's LOCATE (row, column, cursor, raster, raster). I think the ACCEPT AT (row, column) SIZE() VALIDATE() to be more logical and easier to use than GWB's methods. I even like the TI's CALL KEY much better than GWB's INKEY$. It gives more information and is more powerful.

(3) Here is the number one unequalled way that TI XB has it all over GWB: The ability to write truely structured programs by writing your own subprograms that you can CALL by the name you give them. For Example: CALL INTERESTCALC (PRINTAMOUNT, RATE, PERIOD) is tons more meaningful than GOSUB 1300. This alone weighs so very heavily in favor of XB, that for many 99/4A (and 9640) users it may well be the deciding factor in making their judgement of which version of Basic is the better.

So I suggest, again, that some time when you are near one of the very large book stores that have an extensive computer section, you peruse any GWBasic manual. I suspect you may just come away feeling much better about TI's Extended Basic.

LASERS AND THE TI

------------------------------------ by Deanna Sheridan

from the Hoosier User Group newsletter, May, 1993

With the cost of laser printers falling almost daily, you may want to investigate the feasibility of using one with your TI. I have looked for a comprefhensive review of laser printers in newsletter for the past two or three years, and not one article has appeared.

You have all seen what MArty SMoley can do with his aser printer and the TI, but most of us WOULD NOT go through what Marty does to get his printouts. Since none of the TI software has built in printer commands for lasers, Marty has to write his own.

I just purchased a HP Laserjet IIIp and thoght that my experience might help someone els decide if there might be a laser intheir future also. I knew that you could print text fro the TI through funnelweb, but was concerned that I would not be able to use my graphic programs. The paser prints at 300 dots per inch, while most dot matrix printers user under 200 dots per inch. Thus, all you would get is garbage.

If you read the advertisements for lasers, you will see that many of them have several printer emulations. That means that if it is emulating a Diable 630, or an Epson FX, or an IBM Proprinter, that the printer commands fpor those printers wil be accepted by the laser and printed as though they were that machine.

Many of the newer dot matrix printers have a variety of fonts built in. My XR-1000 has Courier, TW Light, Sans Serif, Cinema, and several more that I can shoose from the front panel of my printer. Even though most laser printers advertise 14 basic fonts, they will not be different fonts that you are used to with your dot matrix. These 14 fonts are basically variations of one font ... Courier, ie courier 10 pt, 12 pt, 16 pt (condensed as we know it), in regular, bold italic, etc.

Thus, this is the only font you will get when you are in said Epson, Diablo, Proprinter, etc. mode. The ommands for condedsed, enhanced, double wide, etc. wil be interpreted and printed as you usually expect.

Thus, when considering a laser, the first thing that you will want to know is whether it has built in Epson emulation or whether a seperate card can be purchased to emulate an Epson. With this feature, you can use PAGE PRO, TIPS, etc. without any modifications to the program.

There are several laser printers advertised between the $600 and $900 range, but the cheapest in price may not be the least costly in the long run. If the Epson emulatin is not built in, the cartridge will cost about $100. A lot of these low-end printers only have 512k memory.

A laser printer is a "page printer". All of the data necessary to print an entire page is sent to the printer before it prints anything. Thus, if tere is not enough memory in the printer to hold the whole page, not all of it will print. When reading articles on lasers, they say you shold have at least 1 meg in order to print a full page of graphics. I don't know how many of us would be printing a page of high density graphics and nothing else. But it is some thing you should consider.

The cost of aser memory varies widely from printer to printer. You can add a meg to your PC for about $50, but it will cost you two to three toimes that for your printer, depending on the make.

Another cost to consider is the imaging process the laser uses. Some use a toner/drum combination which is standard in the HP laserjets. Others require you to replace the toner and drum seperately. This is not dome each and every time you need toner, but every so manythousand copies. Some use the HP toners and others use their own brands which are not as plentiful or as cheap.

I seriously looked at the Epson Action Laser II. This is a very basic inexpensive printer. It had Epson emulation built in, and uses Hewlett Packard font cartridges (which you could use for text on the TI). The print speed on this machine is rated at 6 pages per minute which is fast for a low-end printer. Standard memory os 412k and additional memory was about $100 for one meg and $150 for 2 megs. The second meg si sometines cheaper because it goes on the same board as the forst. However, I started to get second thoughts when I found out that the toner cartridges were about $100 each and the drun had to e replaced at 130,000 (I may die before I reach that many tough) at a cost of about $150. The street price on this printer is a little over $650.

Since I would be using my laser in both my TI, and my DOX machine, I looked at a machine I probably would have passed up if I only had my TI. This is a TI microlaser which was highly rated in a PC Magazine review. However when I went out to CompuADD, the only dealer in this area, they did not have one in did not know when they would get one, and couldn't tell me how much an Epson font cartridge would cost or where I could get one. Guess you know they lost that sale in a couple of minutes.

Another machine I would definitely have considered if I were buying for only my TI, would be the Star Laser 4. This comes with Epson emulation, 1 meg of memory, uses HP standard toner cartridges and font cartridges. Most dealers are including a 25-font cartridge in the price which is about $799. It is a 4 page per minute printer. However, it uses something called RISC processing which sends the data out to the printer faster.

Because I intended to use my laser for business as well as pleasure, I chose the Laserjet IIIp for that very reason. It is probably a little more expensive than most of you would feel free to sepnd, and you might choose instead the laserjet IIp which has a street price about the same as the Star printer.

We were getting a Laserjet III at work and what I did at home would be compatible with what I do at work without any modifications. The only noticeable difference for me is the print speed. Of course, it is not nearly as large, with smaller paper trays, but I am finding it adequate for a home printer. The Laser III and up family supports scaleable typefaces, filled fonts, and most of the fancy tricks you see Marty do with his Canon laser. However, just as with Marty's printer, the software drivers for the popular DOS packeages do not yet take advantage of these features. If I am to use then, I have to learn an entirely new language called PCL and have a software program that will let me insert my own printer codes. It comes with 1 meg of ram standard. I added a second meg and got the Epson cartridge which added about $200 to the street price. Recently Micro Center advertised cartridges for the IIp and IIIp at $50.00. Just as diskdettes and ribbons became chiaper when the volume of users increased, we should see toner decrease also. My toner cartridges are supposed to be good for 1,000 copies. After you get a couple of cartridges, you can always have them remanufactured. I am told that the cost for this is about $40.00. When cartridges were selling for $75 to $100 this was a sizeable savings.

I wrote this article using Art Gibson's Newsletter Printer program and the Epson Emulation cartridge on my HP laserjet. I recently purchased a scanner and scanned the picture of a Laserjet from an ad in the Computer Shopper. I converted the scanned image to a Mac file and sent it over to the TI where I converted it to an instance. I used TI-Artist to add the text above and below the picture.

That sounds like a lot of work when I could have just printed it in two columns from Word Perfect with hith resolution graphics and a fancy font. We all know that Word Perfect can do a decent job of desktop publishing, but not everyone knows the TI is no slouch considering it's age and lack of memory and sophistication.

JUST A NOTE

-- by Goethe

The world is so empty if one thinks only of mountains, rivers and cities; but to know someone who thinks and feels with us, and who, though distant is close to us in spirit, this makes the earth for us an inhabited garden.

FEEDFORTH - MAY '93

--by W. Leonard Tabbs

from pages of the SouthWest 99ers newsletter, Mar 1993

Following up on last month's suggestion for a question or challenge to pose readers, here is an "algorithm". The challenge is to answer if the following program solves the problem. Does it solve it economically? Is there any error in this program? Is there a much easier way to solve the problem?

The program is to accept input whether by user or from reading a file, and then save all items that differ. That is to say, suppose you have an alphebetical list with several items with the same letters. Suppose you are reading a hundred items and you only want to have a list of individually different items. If "ONE" is repeated 10 times, you want to accept the first "ONE" and reject the rest of "ONE"'s untill you come to the next alphabetically arranged item such as "Ondine's" untill it reaches the next alpha order such as "Ongoing". The reason for 2 inputs is when data is read, it must compare the next data item to the previous data item to determine whether it is the same or different. Here the program is listed as User Input so one diesn't have to mess with files untill the algorithm is working. Hope somebody out there can have fun checking this out! There is more than one way this can be done and it will be interesting to see how many ways might be created. The ideal is to have the program as short as possible. The following looks quite long because of the REMS and the User print lines that let you know what the program is accepting and rejecting as entries. The following is listed as for Extended Basic but can be adapted for TI Basic by creating single lines out of the multiple statement lines. IF-THEN-ELSE has to be achieved in a different way in TI Basic -- ask for help in this if you are not familiar with how to program those statements in TI Basic.

There is more than one problem posed by this program. First is the algorithm of getting the program to do what is intended, and within the program the additional algorithms to process item one and otem two (here they are User Inputs), so that they are appropriately accepted or rejected.

```
1 REM (*ALGO) 4-3-93
2 REM for Extended Basic
90 CALL CLEAR
100 PRINT ::INPUT "1: ":A$ :: CT=CT+1
110 A=ASC(SEG$(A$,1,1)) :: A2=ASC(AEG$(A$,2,1)) ::
    A3=ASC(SEG$(A$,3,1)) :: A4=A3
120 IF CT=1 THEN IF A$<>B$ THEN GOSUB 500 :: GOTO 140
130 IF A$=B$ THEN GOSUB 500
140 IF A$<>B$ THEN IF A2=B2+32 THEN GOSUB 500 :: GOTO
    200
150 IF A4>A3 THEN GOSUB 600
160 IF CT>1 THEN IF B$<>A$ THEN GOSUB 600
200 PRINT :: INPUT "2: ":B$ ::CT=CT+1
210 B=ASC(SEG$(B$,1,1)) :: B2=ASC(SEG$(B$,2,1)) ::
    B3=ASC(SEG$(B$,3,1)) :: B4=B3
220 IF B$=A$ THEN GOSUB 700
230 IF C>1 THEN IF B$=A$ THEN GOSUB 700
240 IF B$<>C$ THEN IF B2=A2+32 THEN GOSUB 700 :: GOTO
    490
```

```
250 IF B4>B3 THEN GOSUB 800
260 IF CT>1 THEN IF B$<>A$ THEN GOSUB 800
490 GOTO 100
500 REM * ACCEPT A$ RECORD *
510 PRINT :: PRINT A$;" RECORD ACCEPTED" :: RETURN
600 REM * REFUSE A$ RECORD *
610 PRINT :: PRINT A$;" Record Refused!" :: RETURN
700 REM * ACCEPT B$ RECORD *
710 PRINT :: PRINT B$;" RECORD ACCEPTED" :: RETURN
800 REM * REFUSE B$ RECORD *
810 PRINT :: PRINT B$;" Record Refused!" :: RETURN
```

If it is not clear what the program should do, imagine a list of 700 titles in alphabetical order. Say at least 35 to 45% of these items are duplications. You want a list of the names without having the duplication printed. Perhaps the finished list will only be a single page whereas a list of 700 titles or names will use paper you don't want to waste. This is the purpose of the above problem to solve.

Next are two programs that might stir curiosity if not interest. These are "bare-bones" programs that could be improved cosmetically. The first is a plain and simple file reader which allows reading any DISPLAY/VARIABLE 80 text file. (If you incorporated the INPUT N routine of the second program into this one you could read any DISPLAY/VARIABLE file).

The second program (DV28TODV80) will change a DV/28 file to DV/80 file (but does not provide carriage returns). A writer TEXT Editor will jumble the lines if you are not careful editing the DV/80 version. (The PROGRAM Editor of Funnelweb makes it much easier in this respect!). This program could be used in editing files created by TI's THERMAL printer, for instance.

```
1 REM (SEEAFILE) 4-15-93
10 CALL CLEAR :: DISPLAY AT(12,4):"'SEE A FILE'
   PROGRAM" :: INPUT "":K$
100 CALL CLEAR :: INPUT "ENTER FN$,DSC$: ":FN$,DSC$
110 FN$="DSK" & DSC$ & "." & FN$ :: PRINT :: PRINT
    "Read: " ; FN$ :: INPUT "OK? ":K$ :: CALL CLEAR
120 IF K$="Y" OR K$="y" THEN 130 ELSE 1
130 OPEN #1: FN$, INPUT, DISPLAY, VARIABLE
140 ON ERROR 200 :: INPUT #1:A$ ::CT=CT+1
150 IF EOF(1) THEN 210
160 PRINT CT:A$
170 CALL KEY(0,K,S) :: IF S<>1 THEN 200
180 IF K=82 THEN CLOSE #1 :: GOTO 100
190 CALL KEY(0,K,S) :: IF S<>1 THEN 190
200 GOTO 140
210 PRINT CT:A$
220 PRINT :: INPUT "FILE READ.  ANOTHER? ":RF$
230 IF RF$<>"Y" THEN END ELSE CLOSE #1 :: GOTO 100
```

The program below opens an OUTPUT file. Be sure your copying is free from errors! If the program crashes due to any SYNTAX error, or you BREAK into the program (FCTN/4) while it is running to edit for any reason. BE SURE to immediately enter "CLOSE #2" (in command mode) to close the OUTPUT files BEFORE you make corrections! Failure to do this will lose all output data as well as risk messing up the directory of the output file

disk. Examining the data saved before the crash or interruption of the program, can be useful in checking if the program is doing what you want it to do and might offer immediate clues as to what needs correcting. Use the program for short files at first to make sure it is doing what you want it to do -- this will save you time.

If you happen to own TRITON's SUPER EXTENDED BASIC cartridge, did you know you can list a program to either printer or disk in any line length? This is a remarkable and useful bonus for this cartridge. If you intend to list a program for newsletter editing so that it appears in the same form you would see the program listed to monitor screen, you would enter: LIST "DSKn.FILENAME$:28:n1-n2". Where n is your disk number, n1 is the first line of your program, and n2 is the last line of your program. Note the colons needed immediately after the filename and "28 ". These are necessary to get proper listing. If you fail to enter n1 and n2, the program will simply list as a DV/80 file. (If you do not have the Triton SUPER X-B module, Jim Peterson and others have created programs to list or convert your programs tp DV/28 fomat).

```
1 REM (DV28TODV80) 4-15-93
10 CALL CLEAR :: DISPLAY AT(12,4): "'DV28TODV80' PROGRAM.
   THIS WILL RESTORE A DV28 FILE TO A DV80 FILE" :: INPUT
   "":K$
20 CALL CLEAR :: DISPLAY AT(10,1): "THIS OPENS FILE L130 !
   ......" :: INPUT "" :K$
30 INPUT "OUTPUT FILE,DSC2$: " :OUTFL$,DSC2$
40 OUTFL$="DSK" & DSC2$ & "." & OUTFL$
50 INPUT "INPUT FILE DV VARIABLE #: ":N
100 CALL CLEAR :: INPUT "INPUT FN$,DSC$: " : FN$,DSC$
110 FN$="DSK" & DSC$ & "." & FN$ :: PRINT :: PRINT "Read:"
    : FN$: :"Saving: " :OUTFL$:"" :: INPUT "":K$ :: CALL
    CLEAR
120 IF K$="Y" OR K$="y" THEN 130 ELSE 30
130 OPEN #2: OUTFL$, OUTPUT, DISPLAY, VARIABLE 80
140 OPEN #1: FN$, INPUT, DISPLAY, VARIABLE N
150 ON ERROR 230 :: LINPUT #1:A$ :: CT=CT+1
160 IF EOF(1) THEN 230
170 PRINT CT:A$
180 PRINT #2:A$
190 CALL KEY(0,K,S) :: IF S<>1 THEN 230
210 CALL KEY(0,K,S) :: IF S<>1 THEN 210
220 GOTO 150
230 PRINT #2:A$
240 PRINT :: CLOSE #2 :: INPUT "FILE DONE, ANOTHER? ":RF$
250 IF RF$<>"Y" THEN END ELSE 10
```

NOTE: The CALL KEY routine above gives you the option of pressing <ENTER> to pause during the program process to view items as they are processed. Also, breaking into the program (Fnct/4) can be done if you wish (in Command Mode only!) to PRINT any variables to check what is going on. When you wish to resume the program, enter CON to continue. (DON'T give the command "RUN"! or you will lose all data processed up to your breaking into the running program.) If you mistakenly enter RUN then immediately STOP the program and DELETE your output file, done easily in ExBasic, by typing in command mode: DELETE "DSKn.FILENAME"). (EOF)

IS IT A STACK OR AN ARRAY

by Rich Gilbertson
Portland WORDPLAY,Sep.1992

Some time ago Mike Calkins put a message on the bulliten board requesting articles for WordPlay.

At the time I was reading about stacking arrays and working on programs that create them. So I thought what better than an article explaining what they are and how they work. It's amazing how such a simple request from one person can get another person to come up with a simplified answer to such a complex question.

A stack is a data structure which behaves like a stack of physial objects, such as a stack of paprer, a stack of cards or a stack of plates.

Stacks are often refered to as PUSH-DOWN STACKS or PUSH-DOWN LISTS. This terminology is intended to make you think of those spring-operated mechanisms used to hold plates in cafeterias. The top plate in the stack is at the level of the counter, and is the only one that is accessible. When a plate is removed, the remaining plates "pop up" and the one under it then becomes accessible. When a plate is added, the plates already on the stack are pushed down and only the new plate is accessible.

This cafeteria analogy gives rise to the terminology for adding an item to or removing a item from the top of the stack. When an item is placed on top of the stack, we say that it is PUSHED onto the stack. When an item is removed from the top of the stack, we say that it is POPPED off the stack.

This analogy if faulty in one respect. When an item is pushed on or popped off a stack, the remaining items don't move up and down like the cafeteria plates do. The only item affected is the one pushed on or popped off.

Items are popped off a stack in the reverse order of being pushed on. Using the cafeteria analogy, that means you can not remove a plate lower than the top plate as you would have to pop off the top plate to get the one lower lower in the stack.

A stack is represented in memory by means of an array and a pointer to the top item of the stack. A pointer is a variable whose value designates a particular item in the array. The array or stack has a size and like the plates in the cafeteria the spring-operated mechanism can only hold a certain number of plates. Too many would be called an OVER-FLOW and too few would be an UNDER-FLOW.

In memory an attempt to PUSH one more onto a full stack is an OVER-FLOW, an an attempt to POP one more off an empty stack is an UNDER-FLOW.

When the value of the pointer is the same as the size of the stack, an OVER-FLOW error condition results from a PUSH when the value of the pointer is 0. An UNDER-FLOW error condition results from a POP.

In Extended Basic the line the number table is stack, the string table and number table are also a stack. The program lines you type in are more of an array.

Have you ever noticed while running a Extended Basic program that the program at time pauses briefly?

This is the result of an OVER-FLOW or an UNDER-FLOW of the

stack. The pause is the program using a garbage collection routine to fix the stack. Some things are just not easy to explain.

Later . . . Rich G.

FOR YOUR INFORMATION

from the VAST newsletter, May, 1993
Interesting stuff from amny and varied sources ......

Want to know how much space is left on some disks, but don't want to run a disk catalog to find out? Type this in and try it. The results can be sent to you printer:

```
100 ! DISK MEMORY AVAILABLE
120 ! by Chick DeMarti 1983
130 !
140 CALL CLEAR
150 PRINT "PRESS (S) SCREEN ONLY" ::
160 PRINT " (P) COPY ALSO" ::
170 PRINT "(ENTER) TO EXIT"
180 FOR ROLL=1 TO 6
190 PRINT
200 NEXT ROLL
210 GOSUB 380
220 OPEN #1:"DSK1.", INPUT, RELATIVE, INTERNAL
230 INPUT #1:A$, J, J, K
240 IF AN$="P" THEN 250 ELSE 270
250 OPEN #2:"PIO" , OUTPUT
260 PRINT #2:" - DISKNAME=";A$:"AVAILABLE="; K ;"
    USED=" ; J-K
270 DISPLAY "DISKNAME -";A$:"AVAILABLE=" ;K;"USES=";
    J-K
280 PRINT 290 PRINT "-ENTER NEXT DISK:"
300 PRINT
310 IF AN$="P" THEN 320 ELSE 330
320 CLOSE #2
330 CLOSE #1
340 AN$="NUL"
350 GOTO 210
360 CALL CLEAR
370 END
380 ! CHOICE FROM MENU
390 CALL KEY(0,A,S)
400 IF S=0 THEN 390
410 IF A=69 THEN 360
420 IF A=83 THEN 440
430 AN$="P"
440 RETURN
```

According to TIPS99, the newsletter of the Puget Sound 99'ers, Lynwood, Wa., here's how much console memory you can expect to have after using one of the four CALL FILES commands:

CALL FILES(1) ..	12,876 bytes
CALL FILES(2) ..	12,358 bytes

```
CALL FILES(3) ..  11,840 bytes
CALL FILES(4) ..  11,322 bytes
```

Obviously, if you need only one open file, you'll have more memory available for programming longer programs.

CALL CHAR characters ...

Here are some ready-made characters that can be used in any Basic or XBasic program. These were formerly published by Rick Kellogg in the MICRO Newsletter in Bloomington, Illinois.

```
SLASHED ZERO
CALL CHAR(48,"0038444C54644438")

RIGHT ARROW
CALL CHAR(48,"000804027F020408")

LEFT ARROW
CALL CHAR(48,"00102040FE402010")

COPYRIGHT SYMBOL
CALL CHAR(48,"003E415D515D413E")

CENT SIGN
CALL CHAR(48,"00083C4848483C08")

PI SYMBOL
CALL CHAR(48,"0000FE2828282828")

CHECK MARK
CALL CHAR(48,"0002020404482810")

SOLID LINE
CALL CHAR(48,"00FF")

UP ARROW
CALL CHAR(48,"081C2A4908080800")

DOWN ARROW
CALL CHAR(48,"00080808492A3C08")
```

The CALL CHAR statements above use zero (48) as the designated character, but any availale character may be substituted.

THREE MORE 2-LINERS AND A LOAD HINT

by Glen Bernasek
TI-CHIIPS, Cleveland

xxxxxx

It's been quite a while since I've written an article. This is because I wasn't quite sure just how a program submitted within an article would come out in print. Well 99'ers, I do want to share some of the recent work I've done on my TI with you. So here goes!

Just one word of caution. My 2-LINERS are fully operational only because they incorporate 'extended line' programming techniques, in that each line is well beyond the four (4) screen line programming default. If you are unfamiliar with the various techniques of extending a progream line beyond four (4) screen lines, then either ask a fellow member how to do this or ask your disk librarian to get you a copy of 2LINERS+3 from the CLEVELAND AREA TI-99/4A USER GROUP library.

The reason I wrote my utility 2-LINERS was that I thought it would be much quicker (and simpler) to have a utility routine available from a "LOAD" menu rather than within a full blown utility program where multiple menu choices and key strokes must be performed to do a "sometimes" simple and straight forward job. The first 2-LINER is called TINYCAT. This is a quick little routine that will list the disk's name, how many sectors are left on the disk and the names of ALL files on the disk. It won't tell you the type of file, but all you wanted to know was what was on the disk anyway. If you need to know more, let the big boys, like DM1000, provide that information.

```
     100 CALL CLEAR :: INPUT "DRIVE  NAME?(DSKn):":D$  ::  INPUT
"<0>-SCREEN  OR  <1>-PRINTER?" :D :: OPEN #1:"PIO" :: CALL CLEAR
:: OPEN #2:D$&".", INPUT, RELATIVE, INTERNAL :: INPUT #2: P$, W,
X, Y :: PRINT #D:P$;" (";STR$(Y);" SECT.LEFT)" : :
     110  INPUT  #2:P$,  Z,  Z, Z :: IF P$ = "" THEN PRINT #D ::
CLOSE #1 :: CLOSE #2 :: END :: ELSE PRINT #D:P$, :: PRINT #D:  ;
:: GOTO 110 :: ! TINYCAT (C)1990 G.W.  BERNASEK
```

Just answer the screen questions, and TINYCAT will do the rest.

The next routine will help organize the monster of a disk library you've collected over the years. TINY/LIB will read a disk catalog (disk name, sectors formatted, sectors left and file names), and transfer it to a disk under an APPEND DV/80 text file called D/LIB. The reason I decided on a DV/80 file format was that this type of file can easily be edited with a word processor. This gives you the best of both worlds: a disk file catalogger and an editable file that can grow as your library grows.

(Please be forwarned, TINY/LIB requires two (2) drives.

```
     100 CALL CLEAR :: INPUT "PUT MASTER IN #1 AND D/LIB IN  #2;
THEN  PRESS  <ENTER>.":A$  ::  OPEN #1: "DSK1.", INPUT RELATIVE,
INTERNAL :: OPEN #1:"DSK2.D/LIB", APPEND, VARIABLE :: INPUT  #1:
P$, W, X, Y :: PRINT #2: P$, X, Y : " "
     110  INPUT #1: P$, Z, Z, Z :: IF P$ = "" THEN PRINT #2: " "
```

```
: " " : " " :: CLOSE #1 :: CLOSE #2 :: INPUT "<FCTN/4  >&""RUN""
FOR  MORE OR <ENTER> TO QUIT.":A$ :: END :: ELSE PRINT #2:P$, ::
GOTO 110 :: !TINY/LIB (C)1990 G.W.B.
```

Once again, do as the screen instructions say, and TINY/LIB will do the rest. If you have more disks to catalog, then just press <FCTN/4> and enter "RUN", or just press <ENTER> to quit.

The last 2-LINER is a handy routine which will copy any D/V 80 text file from one disk to another, USING TWO (2) DRIVES. Once again, TEXT/COPY was created to provide me with a simple and handy means of making a back-up of text files created on a word processor. This program is especially useful when you are about to run ASGARD's SPELL IT! on a document you've just completed. (Remember, ALWAYS back-up your documents if you intend to manipulate them in any way.) This is what TEXT/COPY is intended to do.

```
     100 CALL CLEAR :: INPUT "PUT MASTER IN #1 AND COPY  IN  #2;
THEN  PRESS  <ENTER>.":A$ :: INPUT "ENTER FILE NAME:":D$ :: OPEN
#1: "DSK1.   "&D$,  INPUT,  VARIABLE  ::  OPEN  #2:  "DSK2.:&D$,
OUTPUT, VARIABLE ::LINPUT #1:P$ :: PRINT #2:P$
     110  LINPUT #1:P$ :: IF EOF(1) THEN CLOSE #1 :: CLOSE #2 ::
CALL CLEAR :: INPUT "<FCTN/4>&""RUN"" FOR  MORE  OR  <ENTER>  TO
QUIT.":A$  ::  END :: ELSE PRINT #2:P$ :: GOTO 110 :: !TEXT/COPY
(C)1990 G.W.  BERNASEK
```

As with TINY/LIB, just follow the screen instructions, make sure you have a formatted disk in drive #2 and sit back and watch the blinking lights.

The "load hint" I have for you is for the five (5) sector LOAD routine we all have used from time to time. This routine can be easily modified to list and load any program regardless of size. All you have to do is look at LINE NUMBER 5, and look for "AND C>52:. Change this to read "AND C>47", and the LOAD program will list and load ALL routines that are classified as INTERNAL/VARIABLE 254. The trick is that large programs, classified as I/V 254, begin this classification at 48 sectors rather that after 52 sectors as listed in the LOAD routine. So just change the "52" to a "47" and you'll be in business.

For those of you who are so inclined, a revised Ohio Lottery game and file program, for the new "53" number game, is now available. Just ask your librarian for OHLOTTO/53. It comes complete with a preformatted 53 number Lotto file.

INCREDIFILES

by Wanda Ferguson

from the Pomona Valley 99'ers newsletter, May 1993

Jim and I decided one night to see how many actual lines we could put into one Funnelweb file. The confusion started when we reched 600 lines, knowing that out little machine has 16k or working memory, also knowing that one line contains 80 characters because we had filled each line to capacity with capitol X's. So this equalled 48,000 characters or 45.45 k this was inconceivable.

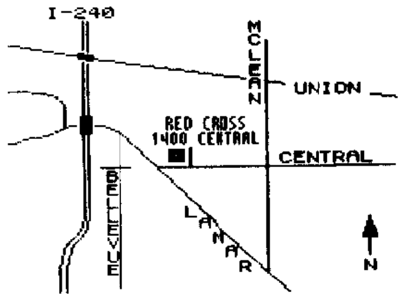
Did the writers of Funnelweb do something to increase the memory capacity? Were we misscalculating k? O.K. find the TI manual and re-chcek our calculations. While I was doing this Jim entered several hundred more lines, now surely with 1,500 lines and 120,000 characters or 113.63k we should be getting an error message letting us know that the file is full. Our calculations for determining k were correct and this only proved to make things more confusing when our line count went as high as 4,350 lines approximately 10 times the size of any other file we've ever created.

O. K. is the computer going to crash when we save this? or try to re-load it into the buffer? Well finally we recieved the first error message of the night it was all in numbers and amounted to the fact that a double sided double density disk was full and the size of the file was above the capacity of the disk. We started deleting lines and trying to save the file, eventually the disk allowed us to save 4,311 lines of 80 characters each, or 344,880 charcters or 326.59 k and the computer will still load, save and edit the file without crashing.

The story isn't over yet. We went to the board meeting and started asking questions. No one could explain it, and most of them didn't believe us. So we recreated the disk that we had reformatted just so we could prove it to everyone.

Well we really did get an incredifile but the experiment failed because if you try to replace even 1 (x) the error message reads TEXT BUFFER FULL SAVE PURGE OR EXIT.

```
Free 0 Used 1440
Size Type / No 1 P
---- ---- ------ -
1438 DIS/VAR  80 U
```



LOCATION MAP

WORKSHOP : to be announced.

PROGRAM BIT - third Thursday

AUG 19th , 1993

MEETING: 7:00pm - Red Cross Building - 1400 Central.

6:45pm - Doors Open

7:00pm - Meeting begins, general discussion.

7:30pm - Demonstration to be announced.

9:00pm - Meeting ends.

9:15pm - Late dinner at location to be announced
at meeting.

NOTICE

Information contained in **Tidbits** is accurate and true to the best of our knowledge. Viewpoints and opinions expressed in **Tidbits** are not necessarily that of the Mid-South 99'ers. We welcome any opinions/corrections from our readers. Articles may be reprinted elsewhere as long as credit is given to the author and newsletter.

GROUP INFO

Visitors and potential members may receive 2 free issues of **Tidbits** while they decide if they wish to join (no obligation) On the top of your label is a code. A Y means you are a member, N means 2 free list, UG means user group and $ means a business. Beside the Y is a date, one year from that date your dues are due. A dollar sign ($) on the label will indicate that your dues are due. The library is open only to members. Library list is $1. Mail order disk library access is $2 for the first disk and $1 for each additional disk - max of 5 disks per month. Order by disk number only. At meetings, library access is FREE if you exchange your disk for ours or $1 per disk for our disks. Send all mail order library requests to librarian's address! Send dues and correspondence to group address.

CALENDAR

MEETINGS:	AUG. 19, (3rd Thursday!)
WORKSHOPS:	TO BE ANNOUNCED

24HR TI BULLETIN BOARD

The 9640 NEWS BBS 300/1200/2400/4800/7200/9600/12000/14400
Hayes. 901-368-0112

GROUP MAILING ADDRESS

Mid-South 99 Users Group
P.O. Box 38522
Germantown, Tn. 38183-0522

LIBRARY ADDRESS

Jim Saemenes
46 Higgins Road
Brighton, Tn., 38011

MEMBERSHIP APPLICATION

NAME ______________________________ |_| $18.00 FAMILY
ADDRESS______________________________________
CITY__________________ST___ZIP_________
PHONE(____)___-______:INTERESTS_______________________________
____
EQUIPMENT,ETC.____

Detach and mail with check payable to: Mid-South 99 Users Group, P.O. Box 38522, Germantown, Tn, 38183-0522.