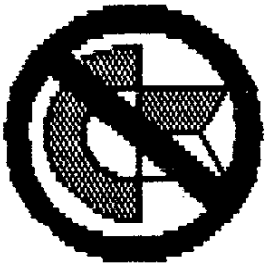
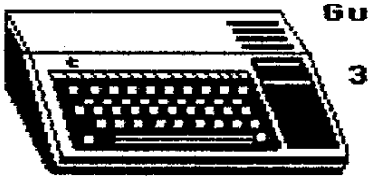
GUILFORD 99'ERS NEWSLETTER


WE'RE NUMBER 1!

SUPPORTING THE TEXAS INSTRUMENTS TI-99/4A COMPUTER



GUILFORD 99'ERS UG
3202 CANTERBURY DR
GREENSBORO NC
27408

TO:

Carmany, Bob
1504 Larson Street
Greensboro NC
27407

George von Seth, Pres. (292-2035) Bob Carmany, Newsletter Ed (855-1538)
Tony Kleen, Sec/Treas (924-6344) Bill Woodruff,Pgm/Library (228-1892)
BBS: (919)621-2623 --ROS

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:Tony Kleen c/o 3202 Canterbury Dr., Greensboro, NC 27408. The Software Library is for dues paying members only. (Bob Carmany Ed)

OUR NEXT MEETING

DATE: May 7, 1991 Time: 7:30 PM. Place: Glenwood Recreation Center, 2010 S. Chapman Street.

Program for this meeting will be a progress report on the UG Library and a brief look at some of the programs we have in it. There will also be a short FORTH demonstration as well. Be sure stop by and see what is available in YOUR library!!!

MINUTES

The April meeting of the Guilford 99er Users' Group was held on Tuesday the 2nd, at the Glenwood Recreation Center on Chapman Street in Greensboro, NC. There were six members present, and one newly registered member. Welcome to our newest member, Roy West. Glad to have you with us, Roy.

The Secretary/Treasurer's report was approved as read. As of 04/18/91, we have $142.91 on deposit.

A discussion regarding the continuation of mid-month classes will be held at next moth's meeting. No classes were scheduled for mid-April as we would all probably be completing our annual IRS misery.

The remainder of our meeting centered on upgrading our club's disk library. Bob led the discussion by showing us four software packages that could be used to catalogue the disks. Each package had its plusses and minuses. CMINDEX is TI-Writer compatible, but needs an external sort and allows 300 or less records per sort. PRBASE allows reporting and has an internal sort and stores 708 records per disk. CATLIB/CATCOM autoloads disk and filenames, has reporting available, stores up to 1200 filenames but is unruly to use.

After all the pros and cons were discussed, we felt that PRBASE would best fit our needs. Bob volunteered to take the library home and start editing. Thank you, Bob.

Everyone is to bring two diskettes to the next meeting. The PRBASE format will be transferred top these disks so that each of us might take a couple of library disks home and catalogue them.

Respectfully Submitted,

Tony Kleen

Cataloging the UG library has turned out to be a bit more of a task than had anticipated. I had hoped to have everything done by this month's meeting. Unfortunately, it doesn't look like it will be finished by then. I do hope to have most of the "hard" work done by then. That is, the labeling and arranging of the library. From there, it is just a matter of typing the program names, descriptions, etc. into PR-BASE. As a result, it won't be necessary to bring any disks to the meeting (see 'MINUTES').

This exercise with PRBASE has produced some unexpected dividends. The question about being able to output the data as a D/V 80 file was a lingering one and I think that I have come up with a solution. After a bit of pondering, I finally figured out a way to generate a D/V 80 file from some of the PRBASE report formats. All that is required is to designate your selected disk drive as the output for your reports. It works fine for 80-character records or multi-line 80-character records but 132-character records are a bit of a problem. With those, you have to be careful or the result is a fractured mess of dis-jointed garbage.

George got his Quest back from Ron Kleinschafer and now has it up and running. In addition to finding the fault (a faulty plated through hole under one of the chips, Ron modified the heat sink under the voltage regulator. With all the work that Ron did on the Quest, George got another good deal. I went over the other night and George now has the Quest as DSK4 and DSK5 and his HORIZON RAMdisk as DSK6. I guess George can learn to live with 3/4 meg of RAMdisk.

Ron said they have had five of the Quests back for various reasons. All of them except George's ended up being assembly faults. Dry solder joints, bad soldering technique and solder bridges were the other minor errors were the main problem. One even came back with the battery pack soldered in backwards. Actually 99% of the problems with hardware projects turn out to be errors by the peron assembling the kit.

I found out how frustrating that sort of thing can be when I put together the Eprom cartridge board that I acquired. There was about 1/64th of an inch clearance between the pin pads and a trace on the underside of the board. Needless to say a couple of solder bridges didn't help it work too well. It took a couple of hours with a multimeter, soldering iron, and a sharp knife to get it up and running properly. The lesson to be learned from all this is to take your time with these hardware projects and check everything carefully!

Back to the Quest modification for a minute. Ron used a piece of Aluminum and made a heat sink approximately 1-1/8" by 1-7/8" that is bent up over the batteries. The surface area is much more extensive than the original. A bit of heat sink grease and everything runs much cooler. I imagine that the same principle could be applie to the HORIZON (or other) RAMdisks but I haven't taken a good look at the construction to see what could be arranged. If anyone is interested, I have the complete drawings. It might be worth the time and effort --particularly if your P-Box is getting full.

ARCHIVING—A HEADACHE?

By: Andy Frueh, Lima UG

A lot of people are puzzled by archiving and how to use Barry Boone's Archiver. What follows is both a reference guide and explanation of Archiver I. It is not meant to totally replace the documentation for this program. Actually, I haven't seen a distribution copy that comes with a set of instructions. There may be hidden features of ArcIII that aren't obvious to me (for example, Disk Utilities by John Birdwell has a feature to figure decimal-to-hex conversions).

What exactly is archiving? Putting it simply, when you archive you take a file or a set of files, and group them as one file then compress them so they take up less disk space. Some software comes archived. These ALMOST always include the archiving program. Examples are Jack Sughrue's PLUS! and the Complete Adventure disk set.

What is the purpose of archiving? Well it started out as a money saver for modem users. It is faster, and thus cheaper, to send 90 archived sectors as 1 file, than 120 sectors for 3 programs. Now it is also a means of backing up disks. You can save each of your disks as a one file, squashed archive. You can specify whether you want compressed files or not. The reason you have a choice is that some unusual files actually take up more space when they are compressed. Another useful application of archiving is when you have programs you want to keep, but don't need ready to use. You can keep archives of all these files instead of taking up disk space.

OK, now that you have the "what", here's the "how". As far as I know, the only archiver is Barry Boone's program. Its operation is completely different from Archiver II. Rather than add new features to past versions, Archiver was completely re-written. It usually contains an XB LOAD program, but may be loaded from E/A. The program's filename is usually ARC1. It can be found on almost all of the bulletin boards, as a commercial version with Geneve utilities, in user group libraries, with other Fairware programs or from the author. Chances are, you can definitely get a copy.

First things first, so get the program loaded. After that, you should see a Fairware notice. Press any key to pass this. You then see a menu. Each menu option is described in detail below.

1) Archive Files - These options are largely self-explanatory. As you may have guessed, this option archives files. Pressing one will deliver a set of prompts. These are "Source Drive (1-Z)". Yes, you can have drive numbered from 1-9 and A-Z. Then comes, "Output Drive (1-Z)". You may use one drive. Archiver will prompt you to change disks when needed. It is highly recommended that you use a blank output disk, since archives may fill or almost fill a disk. Next comes "Output Filename". This is usually the name of the disk you are archiving, or some related heading. For example, a set of D/V 80 articles may be named "ARTICLES". The following prompt is "Pack all Files? (Y/N)". If you answer "Y" then all the files on the source disk are archived. If you answer "N", then when Archiver is working, you are asked "Include filename? (Y/N)" If you answer "Y" then that file is archived, otherwise it is ignored. This is a handy feature if you have programs and files for example, and need them seperated. This process repeats for each of the files on the source disk. The final prompt is "Compress? (Y/N)". Saying "Y" and Archiver attempts to squash each file so it takes up less space. Remember that some unusual file types will actually get LARGER if compression is attempted. When all the prompts are answered, press REDO to correct an error in your answers, BACK to return to the menu, or any other key to continue. When Archiver is done performing any operation, pressing a key goes back to the main menu.

2) Extract Files - This is the opposite of archiving. It will let you pull (extract) files from an ARC file. You are first asked for the source drive. Next you input the source filename. After that, you are asked for th output drive. It must be stressed that the output drive for ALL operations of Archiver should be different than the input drive. You may run out of spave or overwrite a file accidently. Output disks should be blank.

The next prompt asks, "Extract all files?" If you answer "Y" then every

file stored in the ARC file will be taken out. If you answer "N" then when extracting starts, the program asks, "Include filename?" for every separate file in the archive. Again, press REDO (to restart this option), BACK (returns to main menu), or any other key to continue.

3) Catalog Disk - This is fairly self explanatory. Simply input the source drive name. The program will ask if you want a printout. If you answer yes, then you are asked for the printer name. If there are more files than can be displayed, then [more] is printed on the screen and pressing a key advances the screen.

4) Catalog ARC File - If you aren't sure what files are contained in an archive file, than this option tells you. You are asked for the source drive, source filename, and whether or not you want a printout of the list of files.

5) File Copy - This option will copy a file (obviously). Simply supply the source drive and filename, and the output drive and filename.

6) File Rename - Again, this option should explain itself. Give the source drive and filename, then the output filename.

7) File Delete - Supply the source drive and filename.

8) File Un/Protect - You first supply the source drive and filename. You are then asked "Protect?" If you answer "Y" the file is protected. Otherwise, file protection is lifted.

9) List Text File - This will display or print a D/V 80 file. Give the source drive and filename. You are then asked if you want the file printed or not.

10) Load FW - This returns to Funnelweb. Simply give the drive number on which the UTIL1 file is located.

NOTE: When an I/O error occurs, pressing a key returns to the main menu. If you have a Geneve, this is for you. Using a sector editor, find the string 04E08C00 and replace it with D8018C00.

I think that this should get people on the road to understanding archiver. Remember that it is fairware, so if you find it very useful, send the author (Barry Boone) a donation.

[This article/item comes from the January 1991 issue of BITS, BYTES PIXELS (Charles Good, editor), the newsletter of the Lima OH 99/4A User Group, P.O. Box 647, Venedocia, OH 45894.]

CPU PAD MOD

By Neil Quigg Hunter Valley UG

BACKGROUND:

As many interested TI faithful may have noticed, TI in their infinite wisdom managed to waste a considerable amount of available 9900 address map in the area >8000 to >9FFF. One section of this area is dedicated to the CPU PAD RAM. This small block of 256 bytes is decoded into a 1 Kbyte block and responds to the address bases of >8000, >8100, >8200, and >8300 and it is noted in varoius instruction manual that software should be written to conform to the base of >8300.

THE DETAILS:

For the intrepid user club hackers with the do-it-yourself bug and a little practice with the soldering iron, this 256 bytes of PAD can be very simply increased to 512 bytes. All that is required is two 6810 memory chips, some fine insulated wire, a soldering iron and solder, and a little patience.

THE METHOD:

Step 1. Open the console and remove the main circuit board in its tin casing, take care not to damage the keyboard connection.

Step 2. Remove the casing from the board to give access to the electronics.

Step 3. Locate the two existing 6810 memory chips near the cartridge port connector and unsolder pin 12 of each chip.

Step 4. Carefully remove the pin from the circuit board and bend it clear of any contact.

Step 5. Bend pins 10 and 12 of the new 6810 chips so that they will be clear of contact and place the chips in piggy-back fashion on top of the existing 6810 chips. Solder the remaining 22 pins of each chip to the original chip.

Step 6. With some of the fine wire connect pin 12 of the new chips to the hole in the circuit board from which pin 12 of the original chip was removed.

Step 7. Connect pin 10 of the new chips to pin 12 of the original chips and also to pin 1 of either of the system ROM chips on the circuit board. These are the two 24-pin chips located adjacent to the 6810 chips.

Step 8. Visually check all connections and then reassemble the console taking special care to align the metal casing correctly on the circuit board.

FINAL NOTES:

The modification can now be tested by using a debug program or any program that allows direct access to memory locations. If the operation has been a success, you will be able to load different data into memory at >8200 to that at >8300.

The extra PAD memory can now be used for such things as return vector stacks and workspace area and presumably many other handy little things.

SHORT BYTES

DISPLAY MASTER

Have you ever wanted to use your computer in some kind of display? I use to have a booth at our local home show, and it would have been nice to have a presentation using the computer rather than using a slide show. Well, now you can do it with the Display Master software. Display Master was written by Chris Faherty, the whiz kid who did TI ARTIST. Display Master will let you combine pictures and windowed captions to create a sort of slide show.

All you do is use a Dis/Var 80 editor to create a command file and that's about all there is to it. I think the documentation could have been better written, but after you play with it for a little bit, it all seems to sink in. This is where the ARTIST companion products come in real handy..you can combine instances, pictures, use the different typestyles, etc. to create some type of presentation.

Display Master lets you do captions either alone or with a picture and you can also overlay the windows (with the captions) in a Mac-like manner. Probably the greatest thing about Display Master is it's price of only $14.95. Any dealer or TI ARTIST should be able to get it, if not then try Texaments. Display Master really ties the whole TI ARTIST package together!!

A/L NOTES

By Bob Carmany

The first thing to do is to find a book for the beginner dealing with the basic ideas of Assembly Language (hereafter called "A/L"). I discovered much to my dismay that the rather extensive manual that comes with the E/A cartridge assumes a prior knowledge of A/L. Anyway, I finally found a rather elementary text on the subject and decided to spend some time learning to program in A/L -- after all, it was supposed to be easy!

I quickly discovered that books aren't written in logical order. This one was making comparisons between XB and A/L coding and I decided that wasn't the best way to start. You have to understand some basics before you can get tha[illegible]. For example, there was a good deal of discussion of converting numbers [illegible]om one base to another --- a good place to start!

There are three number bases that we have to deal with in A/L programming. I could see that this was going to be fun! Binary (zero's and one's) is the only language that the computer understands. Fortunately, we no longer have to program in binary -- an interpreter does that for us. The other two number bases are hexidecimal (base 16) and decimal (what we all learned in school). I could see that this was getting easier all the time. One of the number bases had already been eliminated. All I had to do was learn how to convert a number from decimal to hexidecimal and vice versa.

OK, let's see what the book has to say! You take the decimal number and divide it by radix 16. I didn't know there was gardening involved in this! I have a whole row of radixes planted out back --- my error, that's radishes! Sorry, back to the task at hand. This is awful! You have to keep track of these "F's" and "A's" when you divide the numbers for the conversions. Anyway, I managed to get through the exercises in the book but it must have had the wrong answers in a couple of places because they didn't agree with my results at all! I could see right off that I had been sold a "blind horse" by Ron and Tony! You know what? I got through the whole chapter and you know what the book said? "The easiest way is to use a decimal to hexidecimal calculator or a computer program to do the calculations for you". Hmmm! I think I just happen to have a program that does that. In fact, it gives you the equivalent in all three bases! So much for that chapter and here is the conversion program.

```
100 ON WARNING NEXT :: CALL CLEAR :: H$="0123456789ABCDEF" :: PRINT "DEPRESS YOU
R ALPHA LOCK KEY": :"PRESS LETTER FOR INPUT BASE": :

110 PRINT : :"D=DEC #  H=HEX #  B=BIN #": : :: CALL SOUND(80,660,6)

120 CALL KEY(0,K,S):: IF S<1 THEN 120 ELSE ON POS("DHB",CHR$(K),1)+1 GOTO 110,13
0,140,150
```

```
130 INPUT "DEC #=":DEC :: IF DEC<-32768 OR DEC>65535 THEN 130 ELSE A,DEC-INT(DEC
-65536*(DEC<0)):: GOSUB 200 :: GOSUB 220 :: GOTO 160

140 PRINT "HEX #=" :: ACCEPT AT(23,7)BEEP SIZE(4)VALIDATE(H$):HEX$ :: GOSUB 180
:: GOSUB 200 :: GOTO 160

150 PRINT "BIN #=" :: ACCEPT AT(23,7)BEEP SIZE(16)VALIDATE("10"):BIN$ :: GOSUB 1
90 :: GOSUB 220 :: GOSUB 210

160 A=INT(DEC/256):: PRINT :"D=";DEC;TAB(12);A;DEC-A6 :: IF DEC>32767 THEN PR
INT " ";DEC-65536

170 PRINT "H= ";HEX$:"B= ";SEG$(BIN$,1,8)&" "&SEG$(BIN$,9,8):: HEX$,BIN$=""
:: A ,DEC=0 :: GOTO 110

180 HEX$=SEG$("0000",1,4-LEN(HEX$))&HEX$ :: FOR I=1 TO 4 ::
A,DEC=DEC+(POS(H$,SEG$(HEX$,I,1),1)-1) (4-I):: NEXT I :: RETURN

190 FOR I=1 TO LEN(BIN$):: DEC=DEC-2 (I-1)*(SEG$(BIN$,(LEN(BIN$)+1-I),1)="1"):
NEXT I :: RETURN

200 A=A/2 :: BIN$=STR$(-(A-INT(A)<>0))&BIN$ :: A=INT(A):: IF A THEN 200

210 BIN$=SEG$(RPT$("00",8),1,16-LEN(BIN$))&BIN$ :: RETURN

220 A=DEC+65536*(DEC>32767)

230 HEX$=SEG$(H$,(INT(A/4096)AND 15)+1,1)&SEG$(H$,(INT(A/256)AND
15)+1,1)&SEG$(H $,(INT(A/16)AND 15)+1,1)&SEG$(H$,(A AND 15)+1,1):: RETURN
```

Maybe this isn't going to be so bad after all. I managed to finesse having to calculate all of those conversions by hand with a short program that I found in my library. Let's see, there is something about registers in the next chapter.

This one starts of with a rather innocent statement. It says that there are three internal registers in the TI CPU --- the Program Counter. the Workspace Pointer, and the Status Register. No worries, mate! This doesn't look to be too difficult! The Program Counter (PC) is a special register that keeps track of the address of the instruction to be performed. After the instruction is performed. the CPU adjusts the address to the next instruction. Right --- the same thing as a line number in XB! Geez, I might have to apologize to Ron and Tony for what I wrote earlier. This isn't too bad so far!

Now for the Workspace Pointer (WP) is another register that contains the address of the program's workspace. Whew! A sentence that says absolutely nothing! OK, a workspace is a memory area of 16 words of memory that are accessed faster than the rest of the computer's memory. Each of these words is referred to as a working register. Aha! I bet that is what they are talking about with those R0 to R15 things in the A/L source code. That means that the Workspace Pointer must point to the first of the working registers -- R0 --- and the rest must follow immediately in memory. This stuff is getting a little more complicated but I think I can grasp the concept.

Now for the last of these registers --- the Status Register (SR). The book says that it holds the individual status bits that are affected by the instructions are executed. Now that makes no sense at all to me. It seems that each of the status bits is affected differently depending on the instruction executed and they can be read by the conditional jump instructions to make the program branch to another routine. That sound like the "IF --THEN"

statement in XB. I guess I'll have to wait until I work with the individual instructions to see which of the 16 status bits they affect. Hey! They even provided a chart:

Name	Abbreviation	Bit Position
Logical Greater Than	L>	0
Arithmetic Greater Than	A>	1
Equal	EQ	2
Carry	CY	3
Overflow	OV	4
Odd Parity	OP	5
Extended Operation	X	6
Not Used	--	7-11
Interrupt Mask	I0-I3	12-15

I've heard of some of these at one time or another but I guess I'll just have to wait and see how they can be tested and used by the various A/L instructions.

Now we are going to look at how A/L statements are actually written. For the time being, the structure of the statements is all I can handle! Back to the book!

It says that A/L statements can have up to four fields. Strangely enough, not all of them have to be present in a statement. In order, they are: Label, Op(eration) Code or Directive, Operand, and Comment. A label, the book says, is only required when you want to refer to a statement from another statement. I reckon it is sort of analogous to a CALL statement in XB -- a group of statements that make up a routine. At least that's the sense of it that I can discern! A label can be from 1 to 6 characters in length and the first character must be a letter (A to Z). A label is the first entry in a statement and each statement can have only one label. Logical enough! I think that I can understand this!

The next part of an A/L statement is the Op-Code or directive. That tells the program what operation to perform on the third field. A simple Directive is MOV for move word. Any of the Op-Codes can be used but they must be spelled correctly (of course). Even my foggy mind can understand that!

The third field is the Operand field -- the item to be manipulated by the Op-Code. There are some rules to follow with this lot as well. Whew! Talk about structured programming!! More than one operand must be seperated by a comma and spaces can only appear in an operand if they are between a pair of apostrophies (the A/L equivalent of parentheses).

The last field is the Comment field. That is a freelance entry to explain what the statement was supposed to do --- and I emphasize SUPPOSED TO! It can extend to the end of the line.

There are some general rules about the form of A/L source code. Each of the fields must be seperated from the other by at least one space. By convention, the fields are generally aligned when they are written. Oh yes, you can add general comments to your programs by using the asterisk "*" at the beginning of the line. It functions as a REM statement in XB.

Although any text editor that outputs a D/V 80 file can be used to create source code, the best that I have found is F'WEB in the ASMode. All of the bugs have been fixed and the word wrap has been turned off. In addition, the code is saved without the tab settings. I find it very easy to use when I'm about to "borrow" a bit of source code from an article somewhere. I haven't gotten far enough with this stuff to write my own code yet!

Ok, let's see what a line of A/L source code might look like. This is for structure only so I have just picked some stuff out of the book:

```
START  MOV  @A,R0 Move A to Register 0
```

Here you have a Label, Op-Code, Operand, and Comment field.

Now, it is time to take a look at a comparison between XB (which I understand) and A/L (which I don't). This example is a bit primitive but it is about all that I can muster right now.

```
100 DATA 1,7                    Defines the two data items 1 and 7

110 READ X,Y                    Assigns the value 1 to X and 7 to Y

120 Z=X+Y                       Adds X and Y to get the value of Z
```

The same program written in A/L would look something like this:

```
X       DATA  1                   Assigns the name X to the value 1

Y       DATA  7                   Assigns the name Y to the value of 7

Z       BSS   2                   Reserves 2 bytes of memory for the value Z

        MOV   @X,R0 Moves X to work Register 0

        MOV   @Y,R1 Moves Y to work Register 1

        A     R0,R1               Adds Register 0 to Register 1

        MOV   R1,@Z Moves the sum to Register 1 replacing the number there.
```

Amazing! I learned how to add 1 and 7 in a program. Watch out Ron Kleinschafer and Tony McGovern--- I'm on the way!!

All you have to do from here is run the source code through the Assembler and you have a D/F object code file that you can LOAD AND RUN. Type in the input filename, an output filename and, after pressing <FCTN-6> the ASSEMBLER EXECUTING message appears. If you are lucky, you get a "0000 ERRORS" message when the Assembler is through.

That would be a new experience for me! Even on stuff that I thought that I had typed in accurately from a printed copy I usually get "double digit" errors! I once typed in about 300 or so lines of source code directly (I thought) from a couple of issues of MICROpendium. It took me two days just to get the typos corrected in that mess!! After all of that, the silly program wouldn't even run. You can imagine how "happy" I was about that! Just wait until I start writing my own programs --- triple digit errors may not be out of the question!

So far, I have to admit that A/L isn't nearly as difficult to understand as I thought that it would be. Of course, I haven't really started to do any substantive programming. I've managed to display a few bits of text on the screen and manipulate some numbers here and there but nothing that would qualify as a program. We'll have to see what the next month will bring!