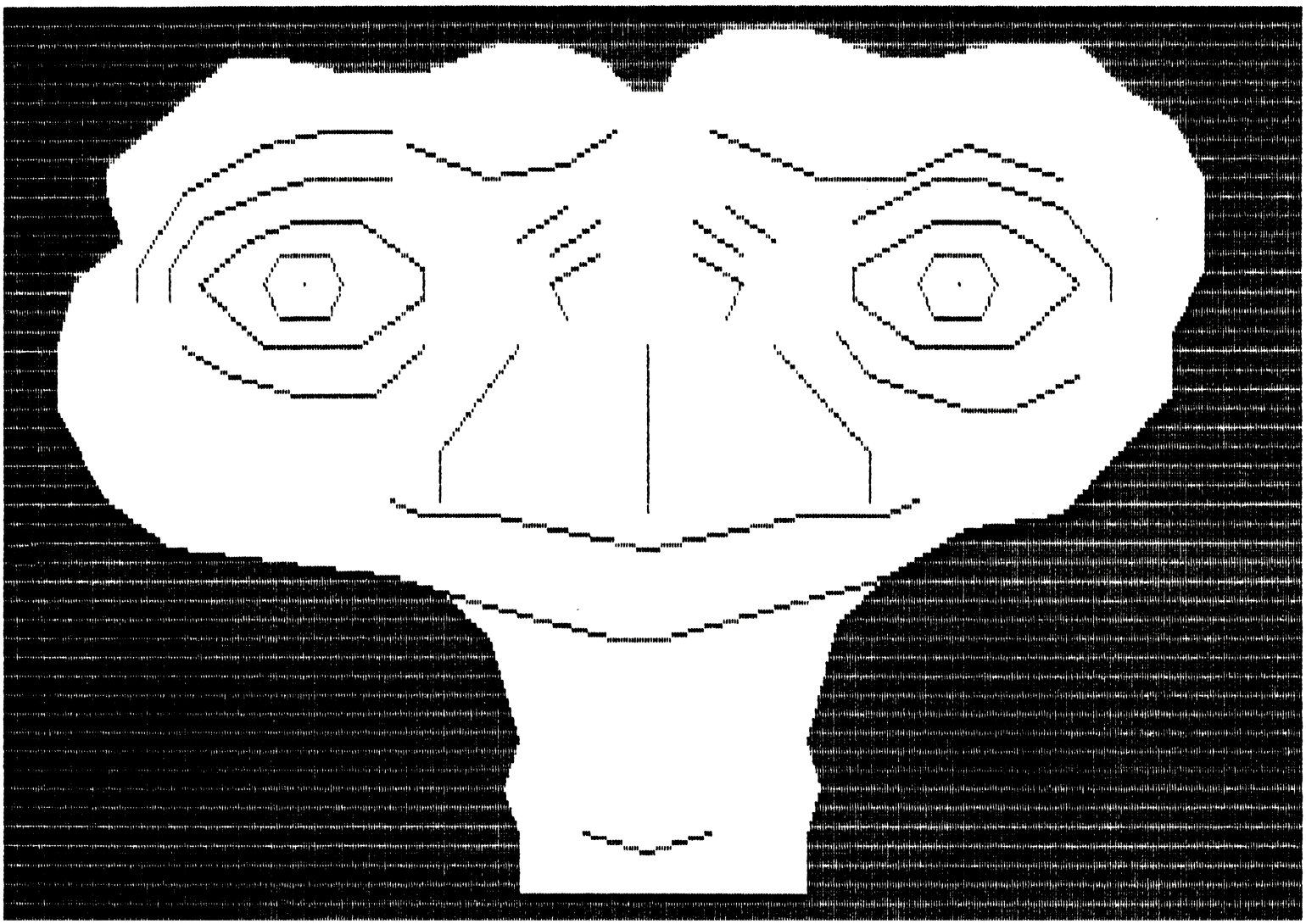
Next Meeting Dec. 4, 1984
7:30pm Details on page 20



E.T. says, "Happy Holidays!"

VOICE OF THE '80

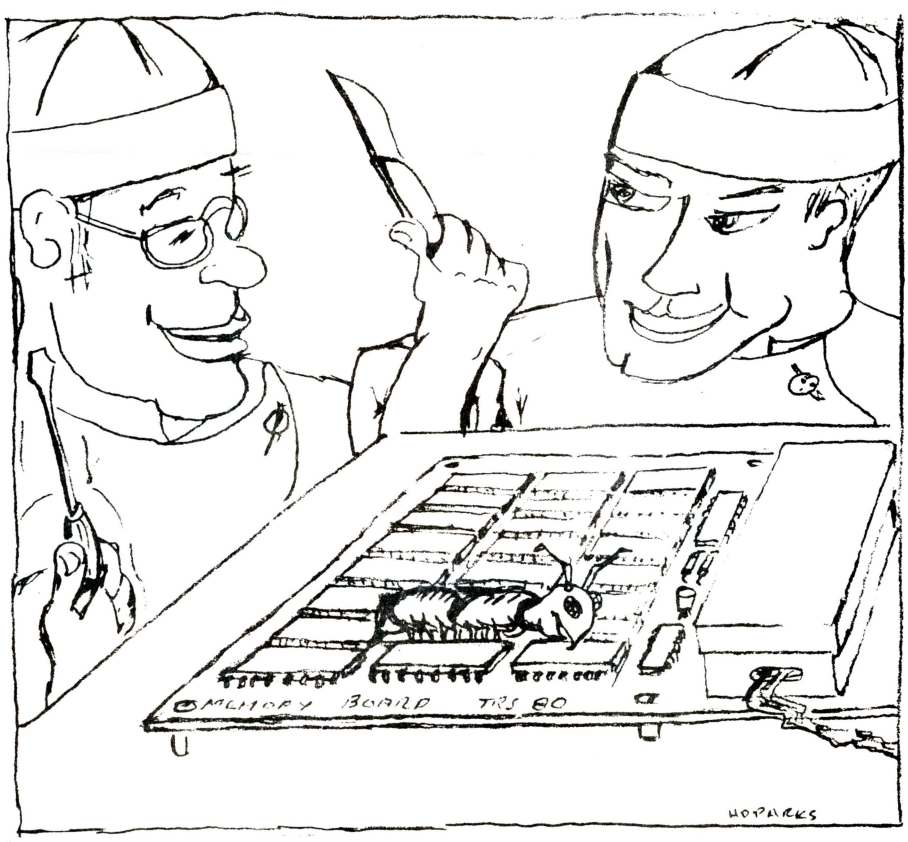
December 1984
Volume 5 Number 7

MEMORY MENDED BY MICRO-MEDICS

By Herman D. Parks

The operating team consisted of Drs. John Roy (chief surgeon), Al Abrahamson (anesthesiologist), Herman Parks (diagnostician). Also attending were Drs. George Phillips and John Bethel (consultants). Dr. Roger Giler, because of his relationship to the patient was present as an observer only. Claire Giler was offered the position of nurse (when you play "Doctor", the girls are always nurses), but she declined.

This was to be a case of the new trend to "surgery in the home" and the patient, a TRS-80 Model III, age 3, lay upon the operating table, formerly the Giler dining table. The patient had been complaining of loss of memory with symptoms of blown programs.



"WHAT SAY, DOC - LET'S FIND THAT BUG IN MEMORY!"

Earlier, Dr. Giler had performed a cat scan by running around the house calling "here kitty, here kitty", but this came up negative. The Abrahamson Test proved more revealing. This consists of entering the following Basic code:

```
1 CLEAR 1
2 PRINT MEM; : GOSUB 2
```

The effect of running this (as described in the Journal of TRS-80 Medicine (AKA "The Voice", May,'83)) is to create an ever increasing stack of incompleted subroutine addresses building downward in memory with protection being moved down with each entry. For reasons still being debated, this is harder on memory than any other known memory test. It will crash at weak chips which other memory tests will pass. The last amount of protected memory printed on the screen when memory fails is subtracted from 38200 (for 48K, approximately - depends on the operating system) to show how far from the top of memory, the weak cell exists.

The first problem the team faced after deciding to "go in" was where to make the incision. Should we cut at the seal and risk charges by the ethics board of Tandy General Hospital? Parks, who was already suspected of questionable practices (where did he get the memory transplants which were to be used?), was for cutting into the side of the patient. Phillips wanted to soak the patient in hot water so the seal could be removed and replaced intact. The incision decision was made and Dr. Roy deftly stabbed through the seal with a surgical screwdriver.

With the patient laid open, the team had to work quickly. The Anesthesiologist, would pull the plug and the Chief Surgeon would replace suspected chips with the transplants. The patient was then returned to life support. Next the Diagnostician would enter the two lines of the memory test and run it. The first transplant of four chips (of the suspected bank of eight) ran to the bottom of available memory indicating that the bad chip had been replaced. Then the operation was repeated with two of the removed chips replaced. When memory again failed, the team knew they had narrowed the problem to one or two chips. Another two rounds failed to isolate a single chip so both were replaced, and the diagnostic was altered to run the test constantly:

```
1 CLEAR 1
2 PRINT MEM; : IF MEM<30000 THEN 1 ELSE GOSUB2
```

This was run for a period of one coke and four cookies. With no failure found, the patient was closed up and discharged.

During the post-operational review, Dr. Bethel approached Dr. Roy and said to him, "I've got this problem - mostly my heart goes 'pitty-pat, pitty-pat, but sometimes it goes 'pat, pitty, pat, pat'". Dr Roy, who could see what was coming, told him flatly, "*I don't do humans*".

New Goodies for the Model 100

By George P. Saladino, Jr.

I went to a computer show here in New York City on October 17th and 18th that was put on by Radio Shack. This was the first show in a long time here in the City that had to do with Radio Shack computers, associated hardware and software. It was quite a show and I saw quite a lot of nice hardware and software that could be used on the Model 100 and other Radio Shack computers. I will not go into all I saw there, but just some of the items I like best.

Radio Shack brought out its new 1200 computer with a hard disk. It is an IBM XT clone. In fact, it looks like an IBM with a different label. The computer looks nice and will run all the software that an IBM XT will run. However, the price difference between it and the IBM is not much. Not as much as Radio Shack states in its advertisement for the unit. IBM has just brought down its price on the XT. Also, IBM has brought out the AT and I feel that Radio Shack is a bit slow and has brought out a computer that is out of style before they market it.

How about more memory for the 100? You could buy the bubble memory from Soundsight and pay $1000.00. However I hear that they have only so far shipped one unit. This is owned by a member of this club, Ben Grundstein.

I am taking a different step for more memory. I bought from PG Design Electronics, Inc., a small unit that fits into the bottom of my Model 100. I will give me 32K more memory and is banked switched. I also bought some software that will allow me to interact between banks. I can have a program working in one bank and have the data generated go the the 2nd bank of memory. I have ordered the unit and software and hope to have it in a week or so. I will let you know more about it in a future article.

Incase you are interested in the unit, the memory unit costs $325.00 and the software is $34.95. You can buy it from-

PG Design Electronics, Inc.
66040 Gratiot
Richmond, Mich. 48062

Their telephone number is (313) 727-2744.

Another item I saw and fell in love with is the "Chipmunk", a disk drive for the Model 100. It is made by Holmes/Amtek and is $499. It take the 3 1/2 inch disks and runs on rechargeable batteries. It has the DOS system in a ROM. The data transfer speed is 10 kilobtes per second. Has a lot of nice features as saving everingthing in memory to disk at once and other added goodies. It connects to the Model 100 in the same socket as the Disk Video Interface does from Radio Shack. However, it has a much improved connector over the one in the Radio Shack unit.

Their address is -

Holmes/Amtek
5175 Greenpine Dr.
Salt Lake City, UT 84123

(CompuServe number is 71676,527)

Lastly I signed up for the Executive Information Service on CompuServe. It has a lot of interesting features. See the sign on menu below.

Executive Information Service
13:19 EST Saturday 03-Nov-84

CompuServe Page EIS-1

EXECUTIVE INFORMATION SERVICE

1 Communications
2 Investments & Quotations
3 Decision Support
4 News
5 Travel
6 Shopping
7 Weather
8 Professional & Technical
9 Consumer Information Service
10 User Information

Enter selection:

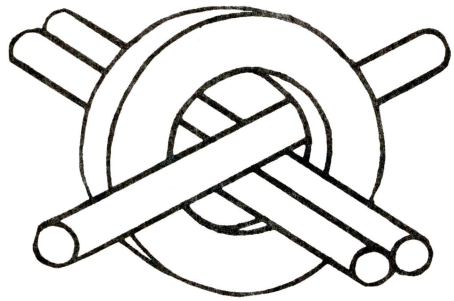
As you can see, there is everything from weather to stock quotes on the service. The information is greater than what you receive in the regular CompuServe service. The price per hour is the same. If you are a member on CompuServe, you just ask for a "Executive Information Service conversion sign up kit" It costs you $39.95 to sign up if you are a member of CompuServe. You will receive a new manual that covers this new service. One catch is that you are billed $10.00 a month minimum. However, I spend over this amount on CompuServe each and every month.

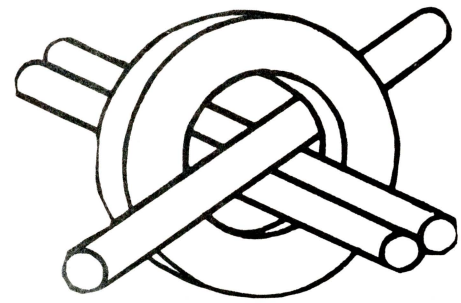
Selection #2 is the most interesting to me. I can obtain not only stock quotations, but also detailed information on the firm I am interested in. Also, one can obtain the price of a stock everyday from several years back to the present.

This service to me is very interesting and if you are interested in stocks and other business news, you should sign up for Executive Information Service on CompuServe.

CORE MEMORY

A history lesson





INTRODUCTION

A core memory plane comes from a computer that has been sold as scrap and destined to be melted down to recover the gold, silver and other precious metals in its circuits. The plane contains a large number of ferrite cores, each of which represent the same single bit position in every word of the computer's memory. The number of bits in the computer word therefore determined the number of planes that are wired together in a stack to make the basic memory unit.

As an example, a large IBM 360 has a 2 byte word (with one parity bit per byte) for a total of 18 bits in its word. Since there is one plane for each bit position in the word, there are 18 of these core planes in the basic memory unit. With 32,768 cores in each of the 18 planes, the basic memory unit contains 32,768 words of two bytes in length or 65,536 bytes total.

CORE WAS POPULAR

Although originally produced by hand-threading individual cores with copper wires, core memory became one of the most cost effective and popular forms of random access computer memory. In fact, it was so popular that during the 1960's random access computer memory was usually referred to as core. The name stuck well into the 1970's even though most real core had been replaced by semiconductor memory. Core memory was not dynamic like semiconductor memory but was instead static. This characteristic increased its popularity since it gave a certain comfortable feeling that the memory contents would be safe even if the power failed. Today's semiconductor memory is still dynamic but the development of special semiconductor memories that can stand-by for months with power from tiny on-board batteries effectively circumvents the need for static memory.

Although popular, core had its peculiar disadvantages. Among these were the power it consumed, the subsequent heat it produced, its relative high price (especially when compared to modern standards) and its destructive read procedure that required overwriting the contents of memory so that it could be read.

THE PHYSICAL ARRANGEMENT

Understanding how core memory works is made easier by looking at how the core plane is physically arranged. Depending on the plane's design, each core has either three or four wires that pass through it. Of these wires, there is always one X and one Y wire. Whether or not the core plane has a single Sense/Inhibit wire is a design decision. This decision involves comparing the cost of the additional logic required to support the single Sense/Inhibit wire to the expense of wiring all the cores with four wires instead of three. Whether the core plane you have has three or four wires passing through each core is easily determined with a magnifying glass. The remainder of the discussion will assume a single Sense/Inhibit wire.

The X wires run from left to right through one row of cores. The Y wires run from top to bottom through one column of cores. Unlike the X and Y wires, which always run from one side of the frame straight through a single string of cores, the Sense/Inhibit wires double back and forth through many strings of cores. By multiplying the number of X wires by the number of Y wires you will arrive at the total number of cores in the plane.

THE PHYSICS BEHIND IT

From basic physics we know that an electrical flow (called a current) when passed through a wire, sets up a magnetic field around that wire in a direction determined by the direction of the flow. Since a core is made of the magnetic material ferrite it can be magnetized, but only if a strong enough current is applied. Applying enough in one direction would result in a clockwise magnetic orientation while applying it in the opposite direction would result in a counter clockwise orientation. By assigning a 1 to one of these magnetic orientations and a Ø to the other we have the basic requirements for binary mathematics.

THE WRITE PROCEDURE

First we will discuss a simplified example that only deals with a single core plane. To write a 1, the computer applies only half the current required to magnetize a core, but it applies this current to two wires, the X and the Y wires that cross each other in the subject core. Since magnetic fields are additive, these currents result in a field strong enough to magnetize only the core that is located at the intersection of these two wires. All other cores that these two wires pass through only have half the current required to magnetize them so they remain unchanged.

Now lets complicate the example by dealing with the stack of core planes in the basic memory unit. Each of the X wires in each plane is connected to the corresponding X wires of the other planes in the stack. The Y wires are similarly connected. In our simplified example we only dealt with a single core plane. Actually a single write would change all cores in the stack if it weren't for the Sense/Inhibit wires. The Sense/Inhibit wires are not connected in the stack but are individually controlled, plane by plane, by the computer. If the

computer is writing a 1 to the stack but a particular core in the stack is set to Ø and is to remain Ø, the Sense/Inhibit wire that passes through that core would be activated with a current to oppose the X and Y currents. This action could Inhibit the write since it would result in a magnetic field that is not strong enough to magnetize that core. If the computer is writing a Ø to the stack but a particular core in the stack is set to 1 and is to remain 1, its Sense/Inhibit wire would have to be similarly activated.

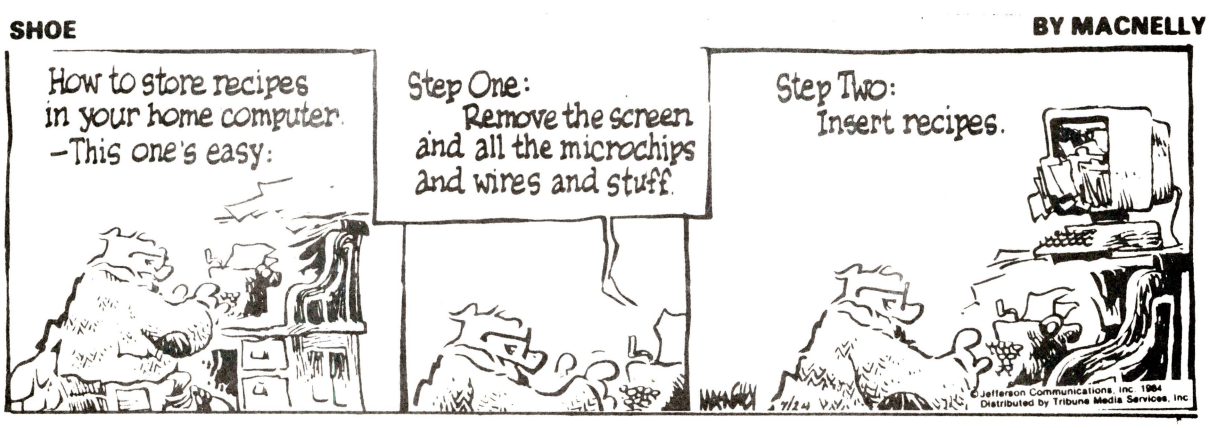
THE DESTRUCTIVE READ

With the write procedure now behind us, we are ready to discuss the more complicated destructive read. To read the stack of cores in a single memory location, the computer first writes a 1 to all of them (it could write a 1 or a Ø but for our example we will use a 1). While it is writing the 1, the computer looks to see if it can Sense a current flowing in any of the Sense/Inhibit wires of that location. If any of the cores in the location being read contained a Ø, writing a 1 would cause that core to change magnetic orientation. This change of orientation would generate electricity and cause a current to flow in the Sense/Inhibit wire that passes through that particular core. After the write and Sense sequence is complete, the computer must then restore the original contents of the memory location by writing a Ø to those cores in that location which originally contained a Ø. (Remember that the first step taken to read a memory location was to overwrite what is in the location with all 1's.)

A core in memory location that originally contained a 1 does not change in magnetic orientation when another 1 is written to it and therefore no current flows in the Sense/Inhibit wire for that core.

CONCLUSION

Core memory and its destructive read have now faded into computer history. As the last aging computers that rely on the ancient technology faithfully execute their last programs, get unplugged and shipped to the smelters, core memory will take its place in computing history along with ENIAC, EAM's card readers, paper tape punches and all the other dinosaurs from the first age of computing.



The Mailbag

Date: Fri Oct 12, 1984 6:26 pm EST **RECEIPT
From: David R. McGlumphy / MCI ID: 181-7759

TO: * Alan B. Abrahamson / MCI ID: 195-8470

Good news! I found a calculator that does decimal / hexadecimal arithmetic and conversions for only $24.95 - at the Radio Shack computer center. It's a new item. It uses two AA batteries to drive an LCD display, and it can manipulate time figures so that you can calculate how many hours (or minutes or seconds) are between two times of the day. It also has a count-down timer from 99 minutes and 99 seconds as well as an alarm, a signal to beep every hour on the hour if you want, and a clock of course. Thank goodness someone came out with some- thing to rival the TI-Programmer which costs about sixty bucks.

Bug report. The telephone dialer program I wrote in assembly will work better (in some cases, read that "will work") if you make the first instruction LD SP,41FEH to set the stack pointer. I use MODEM80/CMD for telecommunications, and by having the dialer's stack pointer set to 41FEH, I can use the "C" command of MODEM80 to dial the phone without having to reload MODEM80. Once I've dialed the number and have heard it begin to ring, I just press <BREAK> which normally exits to DOS from the dialer, but which instead returns to MODEM80. That MODEM80 sure is a smart piece of software.

Besides putzing around on my trusty model 1, I work with an IBM 370/148, (VSE release 3) though my boss may question the use of the word "work". IBM has to be concerned with security of all kinds because users have sensitive data. One of the tools IBM has provided is a command to ERASE a file when it is being deleted. (VSAM files, for those who wonder.) I haven't had to use that function, but I think it writes binary zeros to the disk file before the catalog (directory) entry is deleted. For no particular reason, I started wondering how such a function might be implemented on my machine.

I started thinking in terms of writing an assembly program because I would probably want to go from sector to sector through the target diskette. I thought of all the wonderful things I'd learned about the GAT sector which is the part of the directory that tells which grans have been used. That is, the Granule Allocation Table is changed whenever a program uses a gran of space somewhere on the diskette to write a file or whenever you save a program or file. The more I thought about having to read the GAT and deciphering its contents to determine which sectors to zero out, the more complicated my program-to-be became. I also thought about the multiple extents that show up in directory entries, but discarded that concern because the GAT information should be all I really needed.

I love NEWDOS80/V2. It has so many goodies in it. I can do so much with that operating system. Take DIRCHECK/CMD, for example. It has this neat function that allows you to clean up your directories. It doesn't fix farkled directories, but it does put binary zeros (as opposed to hex zeros?) in places where there used to be a directory entry but now has only remnant

information because the file was deleted. Surely, thought I, such an operating system would have something in it that would do what I was considering. Lo and behold, it does! Almost. By using three DOS commands, I could accomplish most of what I'd thought about doing. And THAT, folks, is a lot faster than trying to write an assembly program.

DIR tells me how many free grans are available. I have five sectors per gran, so five times the number of grans tells me how many unused sectors there are on the diskette. Let's call that value X.

CREATE FILESPEC REC=X is the DOS command that creates a file with "X" records in it, and unless you tell it otherwise, the logical record length is 256, exactly the size of a sector. When DOS creates the file for you, it puts binary zeros in each of the records.

KILL FILESPEC needs no explanation.

The result? All of the initially unused sectors which could have held information like your love letters and which were still readable by someone using SUPERZAP or other utilities now contain binary zeros which may be less interesting than your old love letters.

Remember that I said these three steps would ALMOST cover up your tracks. If you save a BASIC program that uses only three sectors, the remaining two sectors in the gran remain unaffected. They'll still have in them what you left in them when you killed a file that used them. Is it worth figuring out a way to eliminate that left-over information? Not to me, it's not, but I had fun finding out what I've told you. Some people are into ADVENTURE games. This is my kind of adventure.

By the way, an alternative to using the three DOS commands is to format a diskette and then copy-by-file...not much fun if you have only one drive.

One more tidbit. When I format a diskette, I like to have it identified as mine, in case I send it out, and I like to have the diskette-id show up in lowercase letters. If you look in the NEWDOS80/V2 manual in section 5.2 (DOS System Modules), you'll find that the format command is in file SYS6/SYS. Using SUPERZAP's DFS (Display File Sectors) command to look at SYS6/SYS, you'll find the default diskette name in FRS (File Relative Sector) 0CH (12D) at address 3DH. As it comes from the factory, the default is "NOTNAMED". I use SUPERZAP to put in most of my last name in lowercase. The limit is eight characters, so I couldn't put in "Humble Dave, C.D.P.".

A final note. I have an elapsed time indicator on my model 1. Right now it says we've used this infernal machine for three thousand, eight hundred, ninety-six hours. I also have such an indicator in my '75 Dodge Dart, and I found that the car averages almost exactly thirty-five miles per hour. I generally drive right at the speed limit. Look at it this way - at 35 MPH for 3,896 hours, I'd have driven my car one hundred thirty-six thousand, three hundred sixty miles in the same amount of time we've driven my model 1. The difference is that notwithstanding the time I modified the keyboard with a glass of lemonade, the computer hasn't required repairs. Let's see now. On a dollar per hour basis, just how expensive IS my hobby?

So tell me, Ernestine, what have you learned about your 'puter lately?

Exploring With DDT

by Dr. William Parke
Reprinted from Micro Information Exchange, August 1984

INTRODUCTION: In the standard CP/M package distributed by Digital Research, a program entitled Dynamic Debugging Tool may be the most versatile and useful code for its size (4K) ever released by anyone. In this article, we will explore some of the standard and some of the unusual applications of DDT. We will also introduce a few enhancements which you can employ either during the time the program is being used, or as a permanent patch to DDT. If you have avoided DDT because you thought it was only useful to programmers, you have missed a wonderful opportunity for some slick tricks. Did you ever wish to get rid of the commercials in certain programs which you are compelled to read every time the program is loaded? Or did you wish you could change the wording of certain messages to the user coming out of a single disk drive to transfer a file from one disk to another (and avoid the R/O error message you get with PIP or COPY after thinking you had fooled the computer by switching disks)?. Would you like to learn how to recover text which you spent mental anguish composing with your favorite word processor, only to find your working disk is full, or you accidentally pressed the wrong key which exited from the word processor, or your child pressed the reset key? Read on, then.

THE DEXTERITY OF DDT: In the old days, people used DDT to get rid of bugs in the garden. Now, we use a new DDT to debug the imperfect flowerings of the human mind set in machine code. Also we use it to explore, test, and change machine language, stored words, and hardware states. Consider what you can do with its 13 commands (refer to the DDT documentation or the CP/M Quick Reference Card for a list of them). Below we list both common and unusual applications of these commands. The bracketed letters following the application are the commands needed to do the stated operation. If you find one letter commands a bit to cryptic, remember that DDT was written in the early days of 16K micros, when memory chips were a large fraction of the cost of the computer. There was a premium on small efficient code. Some of the advantages of DDT now running in 64K machines depends on its small size. These include the ability to:

1. Change or patch any program or text disk file which fits into memory (about 48K for a 64K machine).
[I,R,S,A,<CTRL>C,save]
2. Disassemble any code into mnemonic instructions. [L]
3. Run any program or any segment of a program in memory, without need for relocation. [G,T,U]
4. Show all CPU registers after any set of program steps. [X]
5. Alter any CPU registers of flags after any set of program steps. [Xr]
6. Instantaneously assemble and insert code at any point in memory. [A]
7. Directly test any I/O ports on the system. [A,G]
8. Directly test any memory location for faulty memory storage.

[A,G]
9. Save any code or text in memory to disk.
[M,<CTRL>C,SAVE nn file.ext]
10. Display and print memory storage in hex and ASCII.
[D,<CTRL>P]
11. Fill any memory block with a given byte. [F]
12. Change any byte in memory to another hex byte. [S]
13. Move any block of code or text to another location in memory. [M]
14. Load both machine code And intel hex files into memory as executable code, at any starting point. [I,R]
15. Convert Intel hex codes into machine code files.
[I,R,<CTRL>C,SAVE]
16. Perform hex addition and subtraction. [H]
17. Recover text typed into some word processor program but lost because of inadvertent exit from the program, such as <CTRL>C or the infamous "BDOS Err: R/O" in accessing a disk for storage, or because of an accidental computer reset.
[DDT,M,<CTRL>C,SAVE]
18. Transfer files from one disk to another using a single drive. (Amazingly, this even works in transfering text files from MSDOS to CP/M-85 on the Z-100) [DDT,I,R,<CTRL>C,SAVE]

These attributes form such an impressive array of talents for such a little program. In the following section, we shall elaborate on some of the less known talents of DDT, starting with those which require a minimal knowledge of machine code.

DISK FILE TRANSFERING: Suppose you have a CP/M system with no virtual disks, and all drives are occupied. To transfer a file to another disk with the CP/M PIP utility, you must reset the system after changing disks. However, with DDT, no disk reset is necessary. The same disk drive can be used to make the transfer. Suppose DDT resides in the A: drive, the file you wish to transfer is in the B: drive and has the name NAME.EXT. From the B: command prompt, type:

```
A:DDT NAME.EXT
```

When DDT loads the file at 0100, note the "NEXT" hex address, such as 2A00. Look up the decimal equivalent of the first byte, 2A hex = 41 decimal. Subtract 1. This is the number of 256 byte pages occupied by the file. Switch to the destination disk in the B: drive. Type <CTRL>C to exit DDT and then, after the B: prompt, type:

```
SAVE nn NAME.EXT
```

where nn is the decimal number of pages to save, 40 in our example. You now have a copy of the file on the new disk.

The same method works on the Zenith 100 running the ZDOS program DEBUG. You may also transfer files from ZDOS to CP/M-85 with this method. Load DEBUG, then the filename (the N command in DEBUG replaced the I command in DDT), load the file with L1000:100 (L replaces the R DDT command), reset the computer, boot CP/M-85, the SAVE nn NAME.EXT, where nn is the page length

of the file. Of course, 16bit machine code files will not run on the 8 bit side, but you may wish to transfer ASCII files or transfer code files to send to another computer with an 8 bit modem program.

RECOVERING LOST TEXT IN MEMORY: Through various faults not of our making, and some which are, it happens that our favorite word processor is yanked from under our feet, leaving us most unexpectedly with some of our best thoughts stranded somewhere in a machine's memory, but exactly where we know not!

The first rule for recovery is DO NOT PANIC. Very often, we lose control of our reasoning when faced with a loss of such great magnitude as having our words flushed down an electronic black hole. Under impaired conditions, we compound the first error by turning off the monster which gobbled up those words. Reloading the word processor is another mistake. Interestingly, DDT can be used to recover text anywhere in memory even if you were not using the CP/M operating system at the time (such as Heath HDOS).

The reasons are fourfold: 1. CP/M loads directly into high memory. 2. DDT is a small program and relocates itself into high memory. 3. You may examine any memory location with DDT. 4. After finding your lost text, you can save it to disk. Follow the steps below. It may be wise to practice them now by purposefully resetting your computer after entering text. The text search operation can be greatly simplified by adding some code to DDT as described later, but this is not necessary.

0. Calm down

1. Reset the computer

2. Boot CP/M

3. Load DDT

4. Use the display command, D1400, to show memory starting at 1400. Repeat the D command until you see your text pass by. (Word processor text buffers are usually quite high in memory)

5. Note the beginning and ending addresses of your text.

6. Move the text to start at 100. For example, if your text started at 5609 and ended at 89AC, use the command H89AC,5609 to find the difference of these addresses (33A3 here). Then use M5609,89AC,100. Convert the hex difference high byte (33 in this example) to decimal (52 when rounded up).

7. Use <CTRL>C to return to CP/M. Then type SAVE nn TEXT.REC. The nn is the number of pages to save, 52 for our example. If you get a disk full error, you may switch to another formatted disk or drive such as B: and repeat the SAVE attempt. But do NOT use any other CP/M command.

8. If you successfully saved the memory block to disk, re-

enter your word processor and edit away the garbage that may be at the back of your file.

EDITING COM FILES: One of the most popular applications of DDT is the editing, reconfiguring, and patching of machine code (COM) files. Now's your chance to remove or change any program commercials, offensive prompts, and unhelpful help messages. You may also be able to make patches to your favorite programs so that the program recognizes the special features of your terminal. Often, the documentation supplied with the program tells you where to patch for customizing your copy.

The technique is to load the program in memory, locate the offensive segment, change it, then save the program back to disk. WARNING: The SAVE command will write on top of any file with the same name! Before starting a patching session, be sure you have a backup copy of the program on a physically-write-protected disk (tape over the notch).

Suppose you wish to change a line or word which appears on the screen while the program MISWORD.COM is operating. Then use:

```
DDT MISWORD.COM   <--load DDT and file into memory at 0100;
NEXT PC 2400          note NEXT PC address to get file size:
                      24hex=36 decimal; i.e. file has 35 pages
-D100             <--display memory from address 0100
-D                <--display next memory block;
-D                   continue... until you see the word at...
-D                   right of hex codes.............
```

Note the hex starting address of the word. For example, one line of the display may look like:
13A0 1040 0808 4102 0042 4144 2057 4F52 4424 .@..A..BAD WORD$

If you wish to change the phrase BAD WORD, look up the hex equivalent of your replacement text in an ASCII table, then use the Substitute command:

```
-S13A7                <--start substituting at address 13A7
13A7 42 20 <cr>       <--make substitution (20 hex = space)
1348 41 20 <cr>       <--follow each hex byte with a return
13A9 44 20 <cr>
13AA 20 <space,cr>    <--exit sub. with a space and a cr.
```

Take care that your replacement fits in the ASCII range of the original message. Do not overwrite dollar signs (24 hex) or nulls (00), as these are used a line terminators.

Now exit DDT with a Control C, and save the new file. If the program had 35 pages, then the operation appears as:

```
-<CTRL>C
A:SAVE 35 MISWORD.COM
```

Naturally, you should test the new version to make sure the patch was not misapplied.

TESTING MEMORY WITH DDT: One of the quickest ways to test a block of memory is to fill it with a given code, then read it back to see if it has changed. this will catch bad cells in the memory chips, and may detect cross-talk between cells. Other faults, such as address line shorts may not be found. By using DDT, you can see the remarkable ability of the human visual system to detect faults in a visual scene. Use the fill command to fill a suspected block of memory with a given byte, such as ØØ. Then display that block with D. You can even fill the whole memory from Ø1ØØ to the bottom address of DDT with one fill command. The bottom address of DDT can be found at address ØØØ5. Use the list command (L) to show this address. E.g.,

```
-L5
ØØØ5 JMP D2ØØ
ØØØ8 ...
```

indicates that the address D2ØØ is the lowest DDT code address, with CP/M above that code. You may then issue:

```
-F1ØØ,D1FF,Ø
-D1ØØ,D1FF
```

You will then see all of your memory flashing before you, loaded with zeros. (To stop and restart display, use <CTRL>S; to exit D mode, use <space>.) It is rather easy to detect a non-zero byte as the display moves. The same test should be run with other test bytes, such as 55 hex and AA (these have alternating 1 and Ø bits). These tests are not as fast as some, but they are certainly graphic.

TESTING I/O PORTS: In configuring a computer system to a modem or printer, one may feel the frustration of nonstandard connectors, pin assignments, and hardware hand-shaking conventions. Some of the mystery can be alleviated by direct communication with the input-output ports on the computer. This can be done with DDT, provided you have some documentation accompanying your computer which tells what communication chip it uses, and which port addresses are decoded. If your system uses memory mapped I/O, then communications is performed by reading from and writing to a data address after checking the status byte at an adjacent address. Direct I/O is more common on 8080 and Z80 machines, usually employing the 8251 or 8250 chip. The Z1ØØ uses the 2661 chip for I/O control. The 8251 takes up two I/O ports, the 8250 requires 7, while the 2661 has 4. Look up the ports decoded for your machine (in range ØØ to FF hex). Each may be examined and written to under DDT by creating a few lines of code while in DDT. Tuck the code into a safe place high in memory (but below DDT itself), so that it will not be overwritten if you reset the computer. To read the port DØ, use the A command of DDT to create a small program:

```
-A8ØØØ
8ØØØ IN DØ
8ØØ2 RST 7
8ØØ3 <cr>
```

Now if you run this code with -G8000, it will stop at 8002 because of the RST 7 (reset #7), which re-enters DDT with all registers of the Z80 just as if they were left by the program. Use the X command to display the registers. The A register will contain the result of the IN operation. To send the byte 0A out to a port E8, insert the code:

```
-A9000
9000 MVI A,0A
9002 OUT E8
9004 RST 7
9005 <cr>
```

and then run with -G9000. Whatever was attached to the port at E8 should have gotten a linefeed.

ENHANCING DDT: The performance per byte of DDT is certainly impressive. However as those who have used DDT know, it's not perfect. As we have seen, it does not have a built in search command. neither does it let you insert ASCII text directly into memory (the substitute command requires hex code). You have no direct way to see what files are present while in DDT. To remedy these shortcomings, I present a patch to DDT itself which adds six new commands in only one extra 256 byte addendum to DDT after it relocates into high memory. The six new commands are:

B	Create a block of ASCII text which may be searched for by the E command
Exxxx	Explore memory in search of a given stored string, starting at address xxxx
Cxxxx	Change memory with ASCII input, starting at xxxx (exit with DELETE key)
Jn	Jump to new default disk n, where n = 0,1,2,... for drives A,B,C,.....
V	View directory at first match to I command file name (use ????????.??? for all files); 128 byte segment of directory loaded at 0800
W	Where's next directory match; returned index gives the 32 byte offset in the 128 byte segment of the directory at 0800

This is as far as we will go this month. We will continue this article in next month's issue, with the implementation of these patches after a quick review.

ZAPS and PATCHES

Reprinted from Northern Bytes, Volume 5, Number 6

NEWDOS 80 Version 2 Model I Patches contributed by Truman Krumholz:

1. To eliminate the double line feed when in the DOS READY mode: Zero byte F0H in sector 0 of SYS1/SYS. It is normally a 0DH.

2. To allow where you can either enter the date and time on power up or just press <ENTER> to bypass: Zero bytes 67H, 68H, 7AH and 7BH in sector 12 of SYS0/SYS These bytes are normally 20H, EDH, 20H, and EDH.

OPTIONAL NEWDOS/80 Version 2.0 Zaps are reprinted from the CINTUG (Cincinnati TRS-80 User Group) newsletter:

These ZAPs to NEWDOS/80 Version 2.0, Models I and III allow the changing of the PDRIVE parameter without writing to disk. The sysntax for use is the same as before except "B" is used instead of "A" in the PDRIVE command. The zap allows the PDRIVE table in memory to be changed without changing the disk record.

********** ZAP 091 ******* 03/02/84 ********* V2M1 ************
SYS16/SYS,02,FB change FE 41 21 20 06 to C3 C0 51 00 00
SYS16/SYS,04,D4 change all zeroes to:
FE 41 12 20 03 C3 F1 4F FE 42 C2 F7 4F 3E 41 12 77 C5 E5 01 00
07 21 E2 4D 71 23 10 FC E1 C1 C3 F1 4F

********** ZAP 086 ******* 03/02/84 ********* V2M3 ************
SYS16/SYS,02,EB change 7E FE 41 12 20 7E C3 CB 51 20
SYS16/SYS,04,DF change all zeroes to:
FE 41 12 20 03 C3 E0 4F 3D FE 41 20 F8 C5 E5 01 00 07 21 D3 4D
71 23 10 FC E1 C1 18 E3 02

The following ZAP allows you to use the Series I Editor Assembler under NEWDOS/80 Version 2.0 on the Mofel III. It may also work on the Model I, but it has not been tried out. Remember that you must have applied ZAP 025 in order to transfer files from TRSDOS 2.3b over to NEWDOS/80 Version 2.0. The function of this zap is to reload the BREAK enable function of NEWDOS/80. If you are interested in patching other TRSDOS programs, the bytes to look for are 3A C9 32 78 44.

EDTASM/CMD,02,00 change 2A 0D 40 to C3 EA 5B
EDTASM/CMD,05,8D change 4C 69 63 65 6E 73 65 64 20 74 6E 20
to: AE 3E C9 32 78 44 2A 0D 40 C3 6B 58

The next patches are for the FORTRAN package and also enable the BREAK key function. They are also for the Model III, and have not been checked out for the Model I. Note that FORTRAN/CMD is also called EDIT/CMD on some versions.

```
FORTRAN/CMD,02,DA change AF 32 1E 53 to C3 A8 7C 00
FORTRAN/CMD,43,54 change 00 00 00 00 00 00 00 00 00 00 00 00
                    to: AF 32 1E 53 3E C9 32 78 44 C3 D2 54
F80/CMD,00,04     change C3 A1 57 00 00 00 00 00
                    to: 3E C9 32 78 44 C3 A1 57
L80/CMD,00,07     change AF 32 15 43 to C3 E6 74 00
L80/CMD,35,72     change 30 00 00 00 00 00 00 00 00 00 00 00
                    to: 3E C9 32 78 44 AF 32 15 43 C3 07 52
```

BACKING UP A COLOR COMPUTER DISK USING SUPER UTILITY PLUS -

This one's short and sweet - use drive configuration settings T3D,35,17 and Super Utility will be able to backup your Color Computer disks on your Model I/III/4. Thanks to Bill Peek of the Indiana Software Group for this information.

MODEL 4P MEMORY EXPANSION by Iam Webb, Saratoga, California is reprinted from the Cabrillo Computer Society Model I,III,4 SIG Technical Notes:

I had occasion to try to expand a Model 4P to 128K. It appears as though there is NO information on the 4P available. All my magazines seem to ignore this machine. Radio Shack is tight-lipped about it and there doesn't seem to be a technical manual available.

To expand the machine to 128K you used about an hour of time and eight 4164 dynamic RAM ICs. I bought mine from DoKay, 2100 De La Cruz Boulevard, santa Clara, California 95050. They cost $5.45 a piece for 200 nanosecond devices which is what I recommend you buy.

Open your machine, insert the memory chips in the eight available sockets AND move the AMP shorting block from E12-E13 to E11-E12.

Test your memory by using MEMDISK from TRSDOS 6.1.2 [SYSTEM (DRIVE=2,DRIVER="MEMDISK")].

If everything checks, you have it. Congratulations. If you care, 128K RAM buttons are available from the Shack for about $1.59 each.

In The News

Excerpts from Popular Mechanics

Phone Forth

If you've ever wanted to experience the FORTH programming language, but didn't want to spend the cash before knowing if you'd like it, get on your computer and modem and call (415) 538-3500. You'll reach an electronic bulletin-board system called FIG-Tree run by the Forth Interest Group. There, you'll meet -- electronically -- with Forthites of every persuasion. Or, if you don't have a modem, call (415) 962-8653 to talk to an actual human.

Hands-off mouse

Mice are big in computers these days -- not the furry kind, but devices that are rolled about on a desktop or atop a special pad. These rolling devices enable the computer user to point at various spots on the screen, or to enter commands without touching the computer's keyboard.

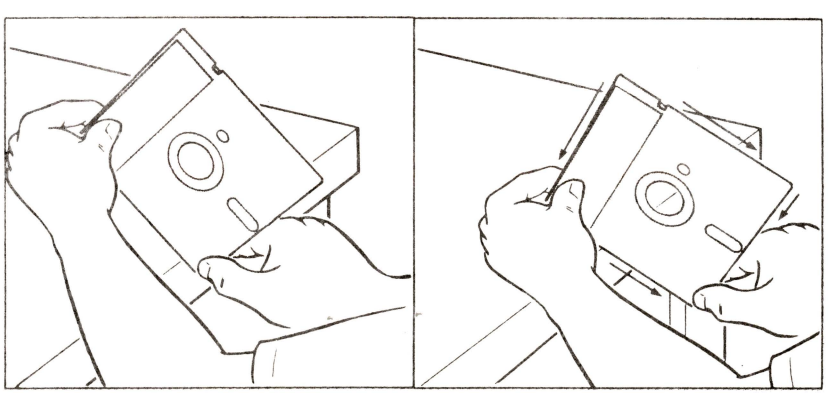
But mice, like all of us, have a problem in that overwork might make them sickly. Moving parts and the accuracy of the rolling surface's circumference all join to contribute to the latest dread of the computerist: mouse failure.

Now comes the solid-state mouse. Announced by Volks Communications of Long Lake, Minn., this new mouse is only a flat pad of transducers. Just by moving your fingertip on top of the pad you achieve the same results as you would using an old mouse.

Of course, a flat pad does not have quite the cuteness ratio of a mechanical rodent. Things could still roll either way.

First Aid For Computer Disks

If a floppy disk won't boot (start up), it may not be moving within its jacket. carefully grasp the disk by two corners and hold it upright on the edge of a desk (see figure 1 below left). Press lightly and draw the edges of the disk along the desktop (figure 1 below right). This will flatten the edges back into shape long enough for you to recopy the data to a new disk.



Your Editor's Ramblings

by Alan Abrahamson

As another year draws to a close, it is only fitting, that someone extend a happy and healthy holiday season to all the membership of the Fairfield County Computer Users Group, their families and all the other clubs that we are in contact with across this continent.

The holiday season, for many, is a time to reflect upon the happenings of the soon passing year. As your editor, I have certainly had my good times as well as the not-so-good times. I can say however, without reservation, that my associations through computing and the Fairfield County Computer Users Group in particular, have only been good times. All of you have encouraged me to continue this newsletter and in fact try to improve its content. I have done the best that time and ability allows. I hope that this experience has been as rewarding for all the readers as it has been for me.

Hopefully, in 1985, FCUG will grow once again in stature. The release of new equipment to the marketplace presents new challenges to the "hacker's" among us. Many new applications packages will be available and problems will abound. So let all of us rejoice in the holiday spirit and continue on the road to microcomputer discovery.

I would be remiss, if I did not offer some special thanks, to particular members and friends, without whose input, this venture would never have continued. If I leave out a name or two, it will only be because of lack of memory and not lack of gratitude. And so,

SPECIAL THANKS GO OUT TO:

Bob Jackson : For leading us through this year of 1984.
Herman Parks: For all the articles he has provided us with.
George Saladino: For keeping up the Model 100 stand.
Dave McGlumphy: For his articles and humor.
Jack Decker: For holding down the TRS-80 information fort.
John Roy: For always keeping my spirits and hardware in order.
Mike Ciferri: For those cover pictures, we all enjoy.
Patrica Brinson: For the miriad of minutes.
??????????: For all the Ask Alan questions.

SEASONS GREETINGS TO ALL

The President's Page

```
*********   NEXT MEETING    *********
***                               ***
***  DATE  :   December 4, 1984 (Tuesday)  ***
***                               ***
***  TIME  :   7:30 pm             ***
***                               ***
***  PLACE:    New Canaan Country School   ***
***                               ***
*****************************************
```

Christmas is just around the corner. Santa is bent over his Model I with hard disk checking the list for the tenth time. There is a bug in there somewhere, but even calls to the local North Pole Radio Shack have been worthless (They are on their eighth manager this year.) Suddenly Santa hears the phone ringing. It is Alan Abrahamson. Roger Giler, Herman Parks and John Roy have found the bug. Christmas is saved thanks to FCUG.

Last meeting we heard a rousing review of structured BASIC and the fantastic word processing program called AllWrite. This month we have a plan to top that with two excellent demonstrations. The plan is MULTIPLAN for the Model IV demonstrated by yours truly who will be spending the rest of November studing the manual for the hidden treasures. Humfrey Udall will demonstrate a real world interface for a manufacturing process. This is a simulation of the real thing because we could not fit the meeting into the factory, but you will get the idea.

The club is as strong now as ever before. Our members are dedicated and enthusiastic. Matter of fact I have even seen some of those new IBM owners coming back to meetings. The rumor mill is that the IBM PC Jr. will not last through March which indicates that EVERYONE is suffering a saturation problem.

I would like to invite anyone who is interested in giving a demonstration to contact me or Herman Parks. The club is interested in demonstrations of new hardware, software and techniques in BASIC, Disk files, whatever.

Bob Jackson

Voice of the '80
Index for December

MEMORY MENDED BY MEDIC Page 1
CORE MEMORY ..... Page 5
Exploring With DDT ... Page 10
In The News ..... Page 18
The President's Page . Page 20
New Goodies, Model 100 Page 3
The Mailbag ..... Page 8
ZAPS and PATCHES ..... Page 16
Editor's Ramblings ... Page 19

The Voice of the '80 is a publication of the Fairfield County Computer Users Group, Inc. Permission to reprint is granted for non profit and non-commercial uses. Credit is appreciated. Submissions should be sent to the Editor at the address below:

PRESIDENT: Robert Jackson
VICE PRESIDENT: Herman Parks
SECRETARY: Patricia Brinson
TREASURER : Robert Gould
EDITOR : Alan Abrahamson
BULLETIN BOARD: (203) 846-6728

Ask Alan HOTLINE: Evenings (6PM-10PM) (203) 866-7883

This Newsletter printed by Sir Speedy Printing Center, Wilton CT

{{{{{{{{ This months cover art by Michael Ciferri }}}}}}}}

Licking, sticking, folding & stapling by: Herman and Jeff Parks

FAIRFIELD COUNTY COMPUTER USERS GROUP
% Alan Abrahamson
10 Richlee Road
Norwalk, CT 06851

FCUG

FIRST CLASS

TO:

First Class