MICROPOLIS USERS GROUP

MUG Newsletter #28 - November 1982

****

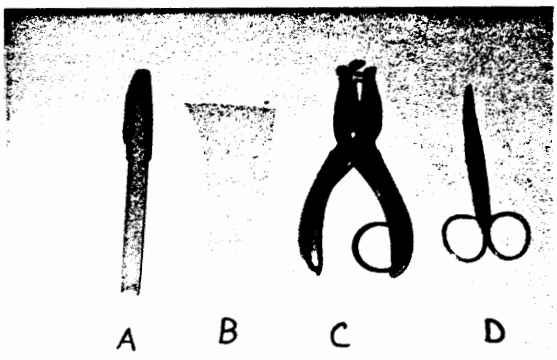
CUT YOUR DISK COST IN HALF

by Lynn Rudow

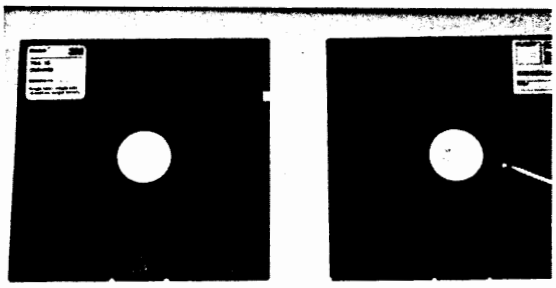
The MOD I and MOD II Micropolis Drives only write to a single side of a disk. As we discussed in earlier newsletters, the reverse side of the disk is most likely as reliable as the front. Performing the modifications shown below will give you a "flippy" floppy disk. Don't confuse the writing to either side of a flippy independently with a single head, with the normal double-sided disk which writes to both sides simultaneously with two heads.

As we've said before, all disks of a manufacturer are produced identically. They go through some testing to validate for various performance requirements. Purchasing a single-sided, single-density disk doesn't necessarily mean that the disk won't support double-sided quad-density. We have found one out of a hundred won't initialize on the reverse side after modification is performed. This is approximately the same ratio of failure as we've experienced with the "good" side.

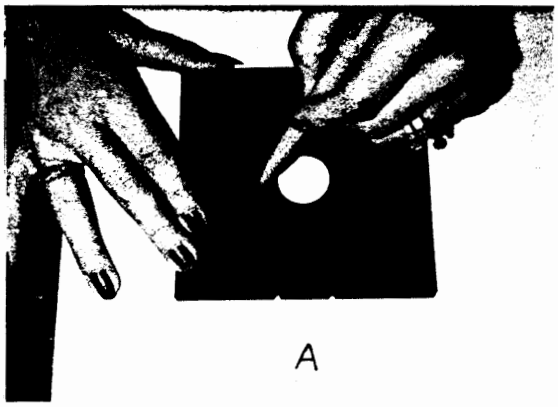
It cuts down your storage space, it halves your disk costs, and takes about three minutes to perform. We make flippies out of all our disks and thought you'd be interested in how its done. While there are kits on the market which sell for $20 to $40, all you need is a few simple tools that you probably have or can easily get.



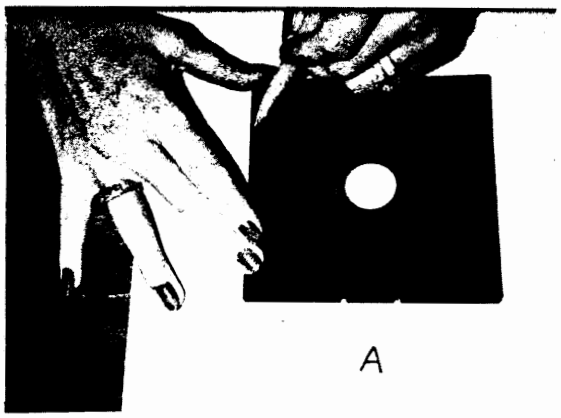
1) Tools you'll need are, yellow felt-tip marker, thin cardboard (we use a business card), single hole punch and manicure scissors. The yellow marker is not a "MUST", but we've found it shows better on the black disk.



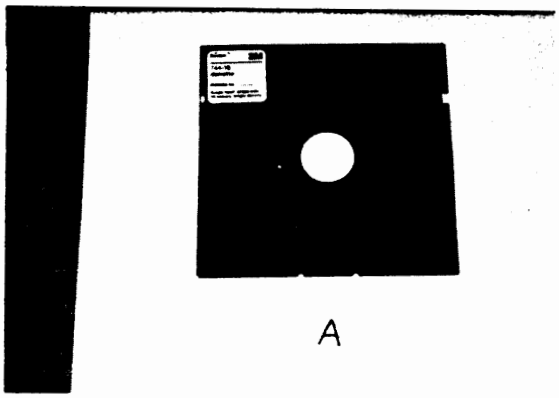
2) Take disk B and align a see-through hole of the disk in the center of the cover hole, as shown in the picture at the lower left.



3) Turn B over, face down, onto A so the hole is on the left side of A. Make sure the disks are lined up. I use the bottom two cut-outs just above the letter A. Holding the disk in place with one hand, mark through the hole of B onto the face of A.



4) Now mark the write protect slot.

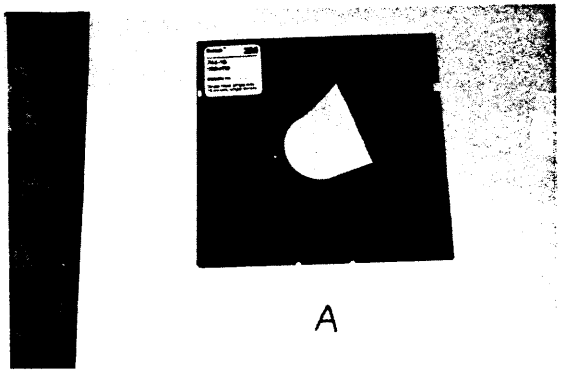


5) Remove the top disk and your disk should look like the above. We used white-out to show the places we've marked, which are the places we'll punch.

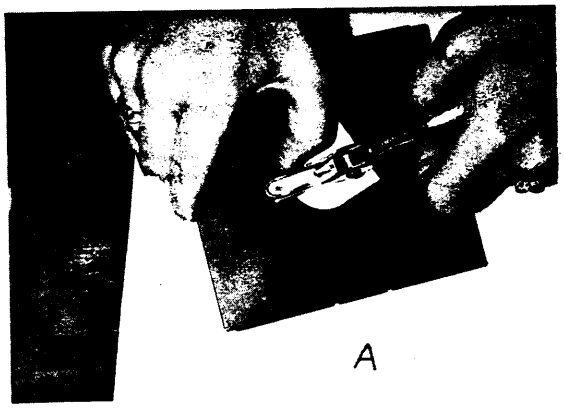
****

SEE THE CHRISTMAS SALE
OFFER AT THE END OF THE NEWSLETTER

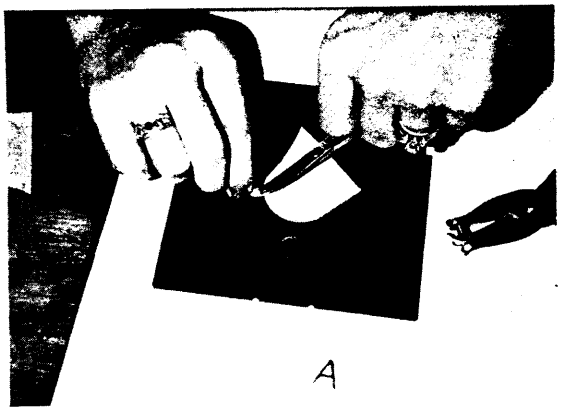
****



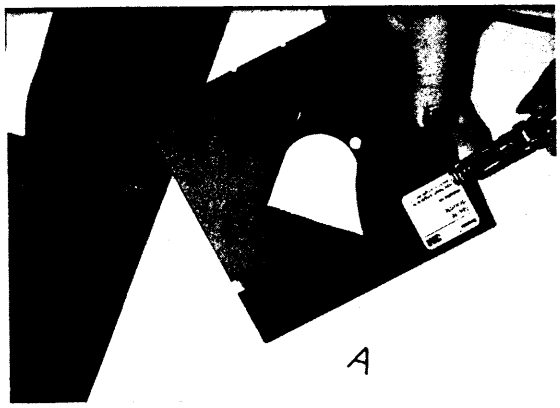
6) Slide the cardboard between the disk cover and disk. This protects the disk when we insert the hole puncher.



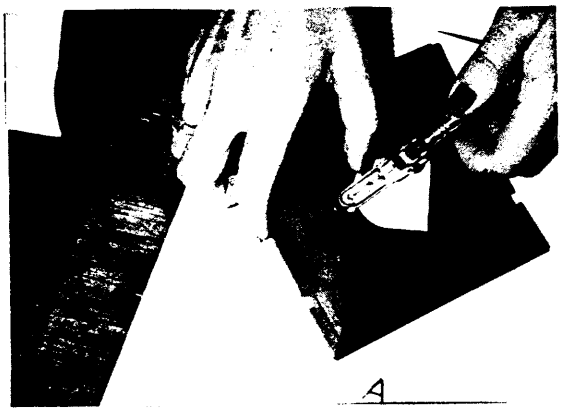
7) Insert the hole punch with the thinner portion next to the cardboard. Try to line-up the hole punch so you are positioned over your marking. Now punch the hole. Tilt the disk so the punched-out part falls out onto the cardboard.



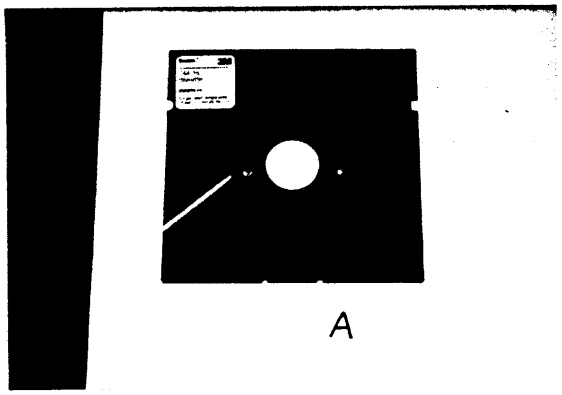
8) Since there is a cotton fiber between the cover and the disk, the hole punch doesn't always cut this. Therefore you need a pair of small sharp scissors to finish the job.



9) Punch the side hole where you see your marker. By-the-way, don't get concerned because you have a round circle and not a square.



10) Now you must turn your disk over and punch the back side following the same procedures in steps 2 through 8, except exclude step 4.



11) You now have two holes, one on either side of the disk. You're through and ready to initialize the back side for use. HAPPY PUNCHING!

.....

WRITING OVER SYSTEM MEMORY

by George Maschino

In the series of articles "Building a Cheap Computer," there was a video driver routine that was to run intermixed inside the RES module of the system. There was a comment that the system would not permit you to load a file below the user program area, as that could overwrite part of the system. In other words, the system protects itself.

Zot went to a lot of work to assemble the code with an offset, and then to load and move the parts of it together into the new system to disk. Most of that work was not necessary as you can load directly into any location in RES or MDOS if you do it right, and if you do not overwrite any necessary code already there.

For example: I have 2 printers on line. Each uses its own driver routine and is connected to its own set of output ports. One of the drivers is in ROM and the system boots up with the RES module set up to call that driver routine. When I want to use the other printer, all I do is load a file called PRINT5. This is a file that loads the needed driver into upper RAM, and a part of PRINT5 is automatically loaded into RES, changing the LDOUT call inside RES to call the driver just loaded. It works every time. I do not have to use any poke or move or anything. I just load the driver and start using it. Here is how:

In the assembly language version of the driver, there are several org statements. They have the function of telling the assembler where the following code is to be placed. They result in the next code going into the next sector on disk, with a new address in the header where that sector is loaded, and the previous sector will not be a full sector.

At the top of the assembly program is the statement ORG 0DD00H. This is where the first part of PRINT5 goes. They are the routines that comprise the driver. Below it is another ORG statement that gives the location of the lookup table for conversion into the Selectric Code, which is located elsewhere. At the bottom of the routines is an ORG statement that says ORG [ADDRESS INSIDE RES], and the very next statement is CALL 0DD00H. The next statement is the ORG statement that locates the lookup table.

WHAT HAPPENS:
The system checks the load address of the first sector it reads in the file, finds it loads at 0DD00H, and that is ok. It assumes the whole file will load there and does not check every sector as it reads the disk. It does not notice that after the first part, the rest is in the forbidden territory.

If the Video Driver routine in the Cheap Computer series had used the ORG statement as the first line in his assembly language version to send a 1 to 5 or so bytes of needless code into a ROM of into never never land where there is no memory, then he could have put all the rest of his code wherever he wanted it inside RES, or wherever, as long as he did not overwrite any of the code that RES uses to access the disk. He could even overwrite vital code if he wasn't careful, but the system would have let him.

.....

****
*
* SEE THE CHRISTMAS SALE *
* OFFER AT THE END OF THE NEWSLETTER *
*
****

MICROPOLIS BASIC ACCURACY

by Morris Barwick
1529 Monaco Dr., Slidell LA 70458

An article on page 22 of the April 1982 Microcomputing caught my eye. The point of the story was that a lot of the common Basics make mistakes. The following program was the object of the article.

```
10 FOR C=1 TO 100
20 IF SQR(C)<>INT(SQR(C)) THEN 40
30 PRINT C
40 NEXT C
```

The program finds the ten perfect squares between 1 and 100. According to the article, and Apple and a PET can only find six perfect squares. I ran the program on a TRS-80 Model III and also only get six answers. Microsoft Basic, the "standard" of the industry, finds nine correct answers.

Guess what? Micropolis Basic (Version 3 and 4) and BASIC/Z get all ten answers correctly.

.....

UNREADABLE FILE NAMES

by Buzz Rudow

If, under Micropolis Basic, you try to name a file with an embedded blank, the system normally stops you. For example, the statement:

```
OPEN 1 "N:FILE ONE" (blank between 'E' and 'O')
```

gets a response of:

```
INVALID FILE NAME
```

However, sometimes you can get in trouble, either in your own program or with commercial software which use the RENAME function. If you:

```
OPEN 1 "N:FILENAME"
PUT 1 "TEST"
CLOSE 1
OPEN 1 "FILENAME"
RENAME(1)="FILE BACK" (blank between 'E' and 'B')
CLOSE 1
```

you'll find it works. That is, there are no illegal statements. Now, however, if you try to reopen the file:

```
OPEN 1 "FILE BACK"
```

you get:

```
INVALID FILE NAME
```

This problem isn't too hard to solve, but the answer can be very useful. This is especially true if you have just renamed 500 entries of a general ledger file "NOV SAVE", its your only copy, and you're going to have to key them all back in if you can't solve the problem.

MDOS doesn't care about embedded blanks, so you simply:

```
LINK "MDOS"
RENAME "NOV SAVE" "NOVSAVE"
```

A slightly more difficult problem occurs if you happen to name a file with a legal drive designation as the first two characters. There's at least one commercial program that has a rename utility that can get you into trouble. It asks you which drive has the file you want to rename. Suppose you answer "1". Then the program asks for the file name. Suppose it is TEST. Then it asks what the new file name should be. Let's say you want it to be TESTBAK, but you mistakenly think the program also needs to know the drive, and you type in "1:TESTBAK". Whoops! That's what it gets named - 1:TESTBAK.

Now, when you try to access the file by:

```
OPEN 1 "1:TESTBAK"
```

the system trys to open TESTBAK on drive 1, but it isn't there. Its name is 1:TESTBAK. You figure out what the problem is and you try:

```
OPEN 1 "1:1:TESTBAK"
```

You get an:

INVALID FILE NAME

Again, MDOS comes to the rescue. To solve this one:

```
LINK "MDOS"
RENAME "1:1:TESTBAK" "TESTBAK"
      or
RENAME "0:1:TESTBAK" "TESTBAK"
```

if the file is now on drive 0.

.....

CHANGING DISK FILE CONTENTS

by Buzz Rudow

When I originally started looking into the above problems, particularly the second one, I didn't figure out how to do it with MDOS and solved the problem with several of the programs on System/z' DISK UTILITY Package.

I'll use a different example. Suppose you have one byte you want to change in a program, but don't want to reassemble the whole thing. The easiest example to illustrate is to change the Micropolis prompt from MDOS VS 4.0 to MDOS VS 4.2.

Executing the program DMAP produces a list of files and the tracks they occupy. On my particular disk, MDOS was on tracks 03 and 04. Using DDUMP, I read these tracks and displayed the data to the screen. It prints one sector at a time, showing both the HEX and the ASCII interpretation. The Micropolis prompt was very evident as an ASCII string in track 4, sector 7.

Now, I used DEDIT to edit track 4, sector 7. Just pushing RETURN steps through the sector, again showing both HEX and ASCII, and making no changes. When the "0" of "4.0" showed up, I typed "32" (for the HEX representation of "2", though there's a way to use ASCII input). Then I typed "R" to resave the edited sector. Sure enough, resetting the computer and rebooting brought up a prompt of MDOS VS 4.2.

These programs aren't limited to use on MDOS disks. While DMAP won't work on a CP/M disk (because it looks for the DIR file), DDUMP and DEDIT allow you to change CP/M data. The CP/M directory starts on track 2, sector 1, for instance, and you can rename, unscratch, or change user numbers with no problem. The CP/M system is on track 0 and 1, and it can also be patched with DEDIT.

The DISK UTILITY Package (UTL-1) also contains programs for a sorted file listing, for unscratching files, for recovering disks with PERM I/O ERRs, for copying disks with PERM I/O ERRs, for verifying disks, for comparing disks files, and the MDOS to CP/M, CP/M to MDOS conversion programs.

.....

****

* *

* SEE THE CHRISTMAS SALE *

* OFFER AT THE END OF THE NEWSLETTER *

* *

****

CP/M EXTENTS

by Lynn Rudow

A few MUG members have asked me to give my opinion on new software. Since CP/M and BASIC/Z happen to be the latest things I've worked with, here are a few of my thoughts.

BASIC/Z running under CP/M is really nice. It's fast, allows me to have large files on the hard disk, and I do believe Zale must have written it with me in mind. Such as, he tells me when I've left out a THEN, FOR, etc. Anyway, as long as a piece of software does what I want it to do, I tend to think "HEY, it's suppose to do that. So what?". I only speak up when I don't like something. In this case, I don't like trying to print a data file, using a pointer file under CP/M.

In data file structuring, the data is written in records. If you have an alphabetically ordered mail list file, which I do, and you want to print labels in zip order, we generate a zip pointer file. This file consists only of the record numbers of the ordered zip codes in the prime data file. A pointer file might look like 0410 0001 0900 0950 1174 0700 etc., meaning that prime record 400 contains the lowest zip code, and prime record 700 contains the highest zip code.

Now, according to my source (Buzz), CP/M opens an extent (16K block, or 128-128 byte records) for the first label to print. After it prints it, the system looks at the next record number to be printed. If it is not in the same extent, it closes the first extent and opens the new extent. This means if the new record isn't within 128 records of the first, if you're using 128-byte records, - or within 64 records of the first if you're using 256-byte records - you are continually opening and closing extents, which takes a lot of time. This really slows down the printing time. It takes one minute to print 36 labels. Before, running under MDOS, I could print 48 labels in the same time.

I could, of course, speed up the printing time if I reconstructed the file to sequential zip order, but that also takes considerable time - and considerable disk space. Never-the-less, it would solve the problem. Since I do not wish to reconstruct a file every time I add new addresses to it, is there any way to make CP/M not care about closing an extent? If the answer is no, then it seems the real answer is to convert the programs back to good ol' BASIC/Z under MDOS. Buzz says the problem doesn't exist there, and Bob Zale tells me the MDOS BASIC/Z also runs faster than the CP/M one. For those applications that don't require file sizes larger than a floppy, this is the only way to go. Time is money. Guess what Buzz is doing?

.....

'TINY' PASCAL

The Chung/Yuen Tiny Pascal is a subset of standard Pascal. The CP/M version of Tiny Pascal includes the following features:

* Random & sequential disk I/O.
* Compiles completely to 8080 native code.
* WHILE, REPEAT/UNTIL, FOR, CASE, IF...THEN... ELSE, PROCEDURES, FUNCTIONS, ARRAYS.
* Easy to use.
* Recursive.
* Self compiling.
* Integer arithmetic.
* Complete source code provided.
* External routine CALL instruction.
* Inport and outport instructions provided.
* More.

Source code is provided to the entire compiler and runtime library with each diskette. This means that you will have every line of source code used in the Tiny Pascal system. Most of the source code is in Tiny Pascal (the runtime library is in 8080 assembler). You can easily re-compile the compiler. Features may also be added (this, of course, requires some knowledge of compiler design).

Tiny Pascal generated object code is extremely fast. This makes Tiny Pascal very useful as a substitute for assembly programming.

Requires: 36K CP/M, most disks formats available.
From SuperSoft.

.....

ALLEN PROGRAM DEVELOPMENT EXTENSIONS

for Micropolis Program Development System vs. 4.0

Copyright 1981: Thomas A. Ceska and Jeffrey A. Bell

The Allen Program Development Extensions greatly increase the power of MDOS vs 4.0. The programs in this package enhance the Micropolis PDS with the abilities to:

-- Assemble Z-80 instructions using the 8080-com-compatible TDL Z-80 mneumonics,

-- Assemble undocumented Z-80 instructions,

-- Assemble relocatable binary files which are located and linked after assembly,

-- Use macros within assembly language programs, and

-- Debug documented and undocumented Z-80 instructions.

The Allen Program Development Extensions diskette contains the files listed below. All executable files are in 8080-compatible machine code except ZEBUG-GEN which must be executed on a Z-80-based system.

Executable files:

ZSSM-GEN -- modifies ASSM to add extended assembler capabilities.
ZEBUG-GEN -- modifies any version of DEBUG-XX to add Z-80 debugging capabilities.
RLDR -- is a relocatable linking loader.
EXRB -- allows one to examine relocatable binary files.
BATCH -- executes MDOS system commands which are contained in a LINEEDIT file.

LINEEDIT Information files:

ALPHA8080 -- alphabetically lists the mneumonics for 8080 instructions.
ALPHAZ-80 -- alphabetically lists the TDL mneumonics for Z-80 instructions.
ALPHAPLUS -- alphabetically lists the mneumonics for undocumented Z-80 instructions.
MCODE8080 -- lists the 8080 instruction set sorted by machine code.
MCODEZ-8080 -- lists the Z-80 unique instructions sorted by machine code.
MCODEPLUS -- lists the undocumented Z-80 instructions sorted by machine code.

LINEEDIT Source Files:

TEST.S, MTEST.S, TEST1.S, TEST2.S, TEST3.S, and MESSAGES.S are six files which illustrate the use of relocatable binary files and macro calls.

MACRO1.D and MACRO2.D are macro definition files which illustrate how macros are defined.

LINEEDIT BATCH File:

TEST.B is an example of a batch file for use with the BATCH utility.

The LINEEDIT source files provided should be used as examples of how ZSSM, RLDR, and BATCH may be used and how they interact.

.....

BASIC/Z NOTES

Two items of importance for Sorcerer Users need to be mentioned. One, be sure to state a proper MEMEND in the configuration routines. You have the same problem with the Exidy monitor using the upper few bytes of memory as you do with Micropolis Basic. Two, the Sorcerer Speedup patches discussed by John Donaldson (MUG newsletter 18, column 13) must be removed and the original Micropolis code replaced.

.....

BASIC "LOAD AND GO" PROGRAM

Several members have asked about the text that goes with the LOADGO program on MUG Library disk 2. The following is reproduced from Vol. 4 of the Micropolis News.

This program is used to patch PDS version 4.0 of BASIC, so it will begin execution of your BASIC program upon booting your system. Note that in line 260 the word "GO" is the BASIC program name of the file to be loaded. You must substitute your own program name for the word "GO", or create a file named "GO" which will PLOADG your program.

Using your PDS 4.0 system diskette, you will need to follow the steps in section 4.4 of your Micropolis User's Manual to key in, name and save your source file. Next, proceed to section 4.5 of your User's Manual to assemble your program into an object file. When the assembly is completed, insure the file type of the "LOAD and GO" object module is 08 so that it can be loaded from BASIC.

The BASIC interpreter must now be patched. First, create a BASIC only diskette as outlined in section 2.2.7 of the manual. Perform the patches outlined in the Software Information Bulletins #12 and #14 to correct errors in the BASIC interpreter. At this time, if you want to eliminate the EDIT, RENUMBER and MERGE commands, follow the procedure in Appendix G of the manual and execute the features program on the PDS diskette. (See option below to save the shortened BASIC.) Bring BASIC up so as not to destroy the patches you have just made and load the "LOAD and GO" program object module. Basic should prompt "READY" when the load is completed.

All patches to BASIC are now implemented. Resave BASIC as follows:

A) Type OPEN 1 "BASIC": ATTRS(1)=8 (return)
B) Type SAVE "BASIC" 16R2B1, 16R5DFF (return)
   or optionally, if features were removed -
   Type SAVE "BASIC" 16R2B1, 16R5700 (return)
C) Type ATTRS(1)=3:CLOSE 1 (return)

This will allow you to save the modified and patched BASIC that will PLOADG your program.

.....

MUG LIBRARY

Enclosed in this mailing is a complete listing of the MDOS and CP/M disks in the MUG library. Hopefully, this will finally clear up the confusion I created when I reorganized the library.

.....

****SALE****

OK, fellows and gals, time for some high-class marketing stratagies. DAMAN would love to get a few of your Christmas dollars. To influence your attitude, we've decided to have a ****SALE****.

For the period of November 1 to December 15, 1982, you may deduct 12.5% from the prices listed in the accompanying price lists. Buy some games for the kids (or yourself). Casually mention the sale to the spouse. Leave the price list laying around with a few well placed check marks beside the programs you want. Call us and discuss your needs at (205) 883-8113 (the DAMAN office), (205) 881-1697 (an office phone in the house which has an answering device, if we're not here), or on our personal number, (205) 883-2621.

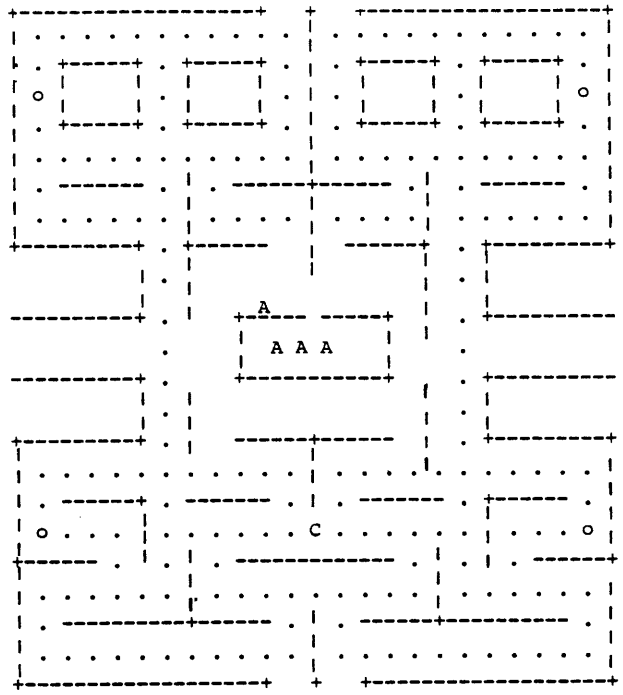
Most Formats Available

We aren't limited to Micropolis format on the CP/M software. If your office is considering a purchase for their Brand-X, we want that business, too.

Even Apple

Though we don't consider ourselves knowledgeable, DAMAN will supply the APPLE games of Avalon Hill, Broderbund, Sirius (and many more), the education programs of EDU-WARE - lots of stuff I know nothing about - postpaid for a minimum of 10% under list. We can make your Apple a full CP/M machine for $659 postpaid. The conversion kit by Microsoft includes their softcard (CP/M and BASIC-80), a 16K RAM card, and the 80-column Videx board. Then you have the capability of running BASIC/Z on your Apple (at slightly additional cost).

CATCHUM



The arcade type maze game by Yahoo Software is finally here.
Requires CP/M

LIST	NORM	CASH
40	20	19

plus $1 shipping to North America, $3 elsewhere.

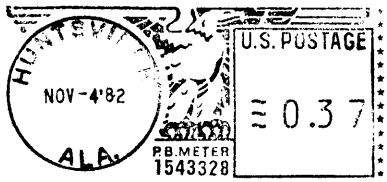
Published Monthly by the MUG
Subscription rates:
U.S., Canada, Mexico; $18/year: Other, $25/year

FIRST CLASS MAIL

FIRST CLASS MAIL

MICROPOLIS USERS GROUP

Buzz Rudow, Editor
604 Springwood Circle
Huntsville AL 35803
(205) 881-1697



FIRST CLASS MAIL