MICROPOLIS USERS GROUP

MUG Newsletter # 21 - April 1982

## POINTERS TO VARIABLE STORAGE

by Burks A. Smith of DATASMITH
Box 8036, Shawnee Mission KS 66208

Like most modern Basic interpreters, Micropolis Basic allocates storage for variables dynamically. That is, storage space is only used when needed rather than always occupying memory. For a given size memory, you can write a long program with little variable storage or a short program with large variable storage and use the same amount of memory.

When you first type RUN, all storage space for variables is cleared and you have a clean slate to work with. Subsequently, your reference to variable names in your program causes memory to be allocated to variable storage in the following way:

A reference to a Real (floating-point) variable causes memory to be allocated in blocks of 26 variables each, representing complete alphabets of letters. If you reference the variable X, Basic allocates storage for the variables A-Z, and any future references to variables in the range A-Z do not require any additional memory. However, if you later reference the variable E7, Basic will allocate another complete alphabet for the variables A7-Z7. Thus, there are eleven alphabets of 26 variables each that could possibly be created by Basic. These alphabets of 26 variables are treated internally by Basic as arrays. Basic only knows the position of the first variable (A) and calculates the position of any other letter. The value of RSIZE is the number of bytes used each real variable, so by knowing the address of the A variable and RSIZE, the address of any other letter variable can be determined.

Integer variables are handled in the same way as real variables. A single reference to an integer variable allocates memory space for all the integer variables A%-Z%. The pointer points to A% and ISIZE holds the number of bytes used by each integer variable.

String variables are allocated one at a time, so there is a separate pointer for each string variable A$-Z$. The first byte of a string variable holds its maximum length and the second byte holds its current length. The remainder of the bytes hold the string itself. SSIZE holds the maximum length of each string unless specifically dimensioned otherwise.

All arrays have a separate pointer and space is only allocated when the array is dimensioned. The internal storage format is as follows: Byte 1 holds the number of dimensions in the array. This is followed by a 16-bit word for each of the dimensions which holds the maximum number of elements in each of the dimensions (the dimensioned value plus one). Byte 1 tells you how many 16-bit words follow it and these words are arranged in the opposite order that they appear in the DIM statement. Finally, there is one additional byte that holds the number of bytes used by each element in the array. The data follows immediately. For a two dimensional arrray A, the data is arranged as A(0,0), A(1,0), A(2,0), ....,A(n,0), A(0,1), A(1,1),.... etc.

It should go without saying that you must know what you are doing before you write a program that does any updating of the data or the pointers, and any program that fools around with data stored by Basic must be thoroughly tested before it is allowed to manipulate any important data. However, once you

(Continued on Page 4)

## FORTH

by Dr. Richard S. Newman
Faculty of Medicine
Memorial University of Newfoundland
St. John's, Newfoundland, Canada A1B 3V6

My interest in FORTH arose out of the need for a fast high-level language. My application was the control of a high speed analog to digital converter and several digital to analog converters. Assembly language coding is tedious at best and the hours spent debugging such code needs no comment. FORTH solved my problems as no other language to my knowledge could.

The differences between FORTH and other languages are many and I don`t propose to go into all of them (I am not really enough of a programmer to do that anyway). FORTH is fast however, and for my application that was essential. Listings 1 and 2 are Micropolis BASIC and FORTH versions of the high-level benchmark program of J. Gilbreath`s (see Byte, Sept. 81). This program calculates the primes from 1 to 8190 without using division. Instead the program uses information about what numbers cannot be prime. On my 2 mHz Sorcerer the FORTH program ran in 15 seconds whereas the BASIC program took 1228 seconds. On this benchmark FORTH is nearly 82 times faster than BASIC and is only 0.7 seconds slower the the equivalent Z-80 assembly lanquage program. ON the Interface Age benchmark (August 1981), Micropolis Basic takes 2251 seconds whereas FORTH runs it in 144 seconds. As a comparison, the IBM System 34, a mainframe, running BASIC took 129 seconds. This is one reason why FORTH is the standard operating language in most of the radio astronomy labs around the world.

FORTH is structured and as such it forces the programmer to use "top down design" and "bottom up programming". Programs assemble from top to bottom and there are no line numbers or GOTO`s which let you make jumps to rescue poor designs. Since each application is broken down into small pieces (top down design), each piece is written and debugged separately (bottom up programming). This also means that changes in a program can be made rapidily (such as in the middle of an experiment) and the program again running in a few minutes.

The core of the FORTH language is quite small which means that machines with a small amount of RAM space can run FORTH. The cost of small size is that standard FORTH systems use only 16 and 32 bit signed integer numbers (software or hardware floating point systems are available if this is a problem). Functions which are common in BASIC like square root, sin, log, string handling, etc. are not available in the core of FORTH although they can be added. The design philosophy was that the applications programmer adds what is necessary to execute the application. The language is therefore not burdened with functions which are not used.

FORTH programs are also conservative of memory because they use very few variables and coding is very tight. FORTH is a stack oriented language so variables go on the stack in much the same way as a Hewlett-Packard calculator. To take advantage of the stack, reverse polish or postfix notation is used, again like the HP calculators. The representation of 2 times 2 in FORTH is 2 2 * as opposed to 2*2 in BASIC. The arguements are entered first and then the operator. Although initially confusing, after you learn to use it you appreciate the fact that no parenthesis are necessary on long math expressions. For example, in BASIC (2+3)/(4+5) translates in FORTH to 2 3 + 4 5 + / . 2 is placed on the stack and then 3. 2 and 3 are added together and the result left on the stack. 4 and 5 are then placed on the stack. 4 and 5 are then added and the result left on the stack. The sum of 2 and 3 (still on the stack) is then divided by the top stack value (sum of 4 and 5). Math is very fast in FORTH because of the stack. Listings 3 and 4 are BASIC and FORTH

(Continued on Page 5)

## WRITING & READING DISK FILES - PART 2

by Buzz Rudow

Last month I discussed the use of the delimeter in disk file structure. One of the nice things about storing data on the disk with delimeters is that you don't normally have to worry about any one field's length. However, you do have to worry about the size of the combined fields in a PUT. If the total size of all your fields exceeds 250, the system gives you an error message. To avoid the excess PUT length, you must keep the number of variables down to a small enough size so that the maximum length of the combined variables won't exceed 250.

Staying with the name and address record as an example, the name-field can be any length.

Buzz Rudow  
or  
Buzz Rudow - C/O DAMAN

I find a couple things wrong with this system.

(1) You can exceed the size of your output form. If you have a 3" label, then a 32 character name doesn't fit (assuming a 10 character inch printer).
(2) It is wasteful of disk space. The N$, A$, and C$ that I used last month won't take up more than 90 to 120 characters. But when Micropolis writes to the disk - that is, you do your PUT - the system uses all 250 characters of the sector you are addressing.

Therefore, one of the first things I tried to do was to compact two logical records into one physical disk record.

If you have 250 characters to use, then you can use 125 for each logical record. A quick check at whether this works is to estimate the size. If I have four lines of data, each being 30 characters, then I use 120 characters.

Indeed, you could input your data that way. My problem was that DAMAN's customers had various requirements, few of which were for a simple three or four line label.

Before I go further, let me say that there is nothing stopping you from storing two logical records on a sector, using delimiters. This discussion, however, is limited to solving the problem by the forcing of fixed field lengths.

Sparing you the details of the various forms I tried before I arrived at my current "standards", this is one of DAMAN's formats for data.

| VARIABLE NAME | USE | SIZE | STARTING LOCATION |
|---|---|---|---|
| A$ | = NAME | 28 | 001 |
| B$ | = ADDRESS 1 | 28 | 029 |
| C$ | = ADDRESS 2 | 28 | 057 |
| D$ | = CITY | 13 | 085 |
| E$ | = STATE | 2 | 098 |
| F$ | = ZIP | 9 | 100 |
| G$ | = COUNTRY | 2 | 109 |
| H$ | = AREA CODE | 3 | 111 |
| I$ | = PHONE | 7 | 114 |
| J$ | = FLAGS | 5 | 121 |
| | | ---- | |
| | TOTAL SIZE = | 125 | |

When inputting data, I check for length and then force the variable to be the size I want. That is
-

```
05 B$=REPEAT$(" ",28): !Makes String of 28 Blanks
10 INPUT "Enter Name: ";A$
20 IF LEN(A$)>28 PRINT "Name must be less than
   29 Characters":GOTO 10
30 A$=A$+B$; !Pads right hand side with blanks
40 A$=LEFT$(A$,28): ! Truncates to the proper size.
```

Each variable is checked for exceeding the maximum

(Continued on Page 6)

## MORE ON THE QUICKENING OF MDOS ON THE EXIDY

by John Donaldson  
Mytton Rodd Ltd., P.O. Box No. 1  
South Melbourne 3205 Australia

In response to several questions about the previous article (MUG: JAN 1982, Pg 13) on speeding up MDOS on the EXIDY, I have tried to explain the principle. While the exact EXIDY changes won't work for others, the principle should be valid for other monitors. I trust that if you find a solution for your system, you'll let the MUG know.

We cannot claim to be experts in the operation of the Sorcerer or the Micropolis, however this is the best explanation we can offer at this stage.

I would suggest that you read this article in conjunction with the Micropolis manual Rev. 8 9/78, pages 4 - 18 and 4 - 19, in particular together with the Sorcerer Software manual 18 pages from the back and the section "Keyboard Quick Check". My Australian edition of the Software manual unfortunately does not have page numbers.

In summary, the patch to the RES module substitutes the Sorcerer Quick Check routine for the @ CBRK routine (4.3.1.3) if the @ CDBRK routine (4.3.1.6) returns a value which is not 0 and not ASCII.

The @ CDBRK routine detects whether and what key has been pressed. It does this twice by first calling the complete Sorcerer monitor keyboard routine and then recognises the character and whether it is 0 or non ASCII (ie. a control key). It then calls the @ CBRK (4.3.1.3) routine.

The patch replaces this second call to the Sorcerer input routine with a call to Quick Check which does the same thing but a lot faster.

The Sorcerer monitor has two keyboard input routines. The Micropolis system has to be written to cover all computers which may not have the Quick Check routine. Hence the Micropolis logic must take the general case rather than the particular in to account.

The patch has substituted Quick Check for @ CBRK routine. Hence the syntax checking only affects the keyboard input routine and substitutes the more efficient Sorcerer Quick Check routine where appropriate - ie. 0 or not ASCII.

We have used this patch for about six (6) months with no crashes or other nasty occurances. We use 2 disk drives with all Micropolis disk commands and extensive interfacing of the computer and the disks with no trouble.

The BASIC Syntax is of course unaffected and all BASIC error messages (Appendix A.) and MDOS error messages (Appendix D.) are all present.

........

## POINTERS TO VARIABLE STORAGE

(Continued from Page 1)

understand the system, you can write assembly language subroutines that add considerable power to Basic programs. For example, I found it a fairly simple matter to write an assembly language subroutine that can sort all the elements in an array stored by Basic quicker than you can say ZAP!

The following are the pointers to the variables as used by my copy of version 4.0 Basic. Testing on a number of different computers indicates that it is probably the same on all version 4.0 Basics.

| | |
|---|---|
| RSIZE | 04C5 |
| ISIZE | 04C6 |
| SSIZE | 04C7 |

| The pointer at | Points to |
|---|---|
| 3421 | A0 |
| 3423 | A1 |
| 3425 | A2 |
| 3427 | A3 |
| 3429 | A4 |
| 342B | A5 |
| 342D | A6 |
| 342F | A7 |
| 3431 | A8 |
| 3433 | A9 |
| 3435 | A |
| 3437 | A$ |
| 3439 | B$ |
| (24 MORE WORDS POINT TO C$-Z$) | |
| 346B | A% |
| **** ARRAYS **** | |
| 3385 | A( ) |
| (25 MORE WORDS POINT TO B()-Z() ) | |
| 33B9 | A$( ) |
| (25 MORE WORDS POINT TO B$()-Z$() ) | |
| 33ED | A%( ) |
| (25 MORE WORDS POINT TO B%()-Z%() ) | |

..........

## FORTH

(Continued from Page 2)

programs which calculate the product of 2 * 2 1000 times. At 2 mHz, FORTH executes in 2.5 seconds whereas BASIC takes 27 seconds.

There is a lot more I could say about FORTH but perhaps it would be more instructive if I mentioned when I use FORTH. I use FORTH for all real time applications (data acquisition, display, control of stimulators etc.). I have also used FORTH for writing plot routines and for cursor addressing of my memory mapped video. When I had to calculate the best way to handle my morgage renewal on the house (our morgages are for at most 5 years at a time) I wrote the programs I needed in BASIC because it had functions which I have not yet implemented in FORTH and at the time I did not have my floating point FORTH. I have also used BASIC to write satistical analysis routines and a program which simulates the electrical activity in nerve fibres. Again these programs required functions not contained in FORTH. All of these programs could have been written in FORTH but at the time it was easier using BASIC than implementing the needed functions in FORTH.

The FORTH I use on my Sorceror is Laboratory Microsystems Z-80 FORTH which costs $50.00 (this includes a manual with all of the FORTH definitions, lots of FORTH programs, a description of the FORTH design, a screen editor and a Z-80 assembler for FORTH). For the floating point software version the price is $150.00. If you really want fast floating point math you can get a version which supports the AMD 9511 on an S-100 board and in addition implements many of the math functions not included in FORTH. CP/M version 2.0 is necessary for all of these systems. Laboratory Microsystems (4147 Beethoven Street, Los Angeles, CA. 90066) can supply FORTH on the Micropolis Mod II format diskette. For a good book on FORTH, I recommend "Starting FORTH" by Leo Brodie which is available from Mountain View Press (P.O. Box 4656, Mountain View, CA. 94040). The price is $16.95.

I have enclosed a copy of the FORTH source for the Users Group (MUG Library Disk 19). To assemble, use FORTHSOR1. It will call up the other three source files as it needs them. The start address is 2B00. Since it uses MDOS calls it must have SYSQ1 and SYSQ2 on drive 0 during assembly. The FORTH module is executable as it stands.

It is pretty certainly not bug-free. I cannot get the editor to work correctly (the editor is not included on the disk ). It will, as it stands, execute FORTH but you cannot store your programs on the FORTH screens. In addition, error messages which normally come from the disk are not present so you have to figure out your own mistakes when it

dosen`t like your statements. Any interested MUGer could have a lot of fun with it, however, and start to learn FORTH while we continue to debug it. Then, as we debug it, we can notify users of the necessary fixes. The complete glossary of the instruction set can be obtained from Forth Interest Group (P.O. Box 1105, San Carlos, CA. 94070, (415) 962-8653). $15.00.

Listing 1
SIEVE Program in Micropolis BASIC

```
1 B=100
2 DIM A%(B)
5 C=0
6 FOR I=0 TO B
7 A%(I)=1
8 NEXT I
9 FOR I=0 TO B
10 IF A%(I)= 0 THEN 18
11 P=I+I+3
12 K=I+P
13 IF K>B THEN 17
14 A%(K)=0
15 K=K+P
16 GOTO 13
17 C=C+1
18 NEXT I
19 PRINT C, "PRIMES"
```

NOTE- "B" in line 1 should be 8190 but that takes too long. If B=100 then it takes 15 seconds to find the primes from 1 to 100 or 81.9*15=1228 seconds for the primes from 1 to 8190

Listing 2
SIEVE program in FORTH

```
8190 CONSTANT SIZE
0 VARIABLE FLAGS SIZE ALLOT

: DO-PRIME
  FLAGS SIZE 1 FILL (SET ARRAY)
  0 (0 COUNT) SIZE 0
        DO FLAGS I + C@
           IF I DUP + 3 3 DUP I +
                BEGIN DUP SIZE <
                WHILE 0 OVER FLAGS + C! OVER +REPEAT
                DROP DROP 1+
           THEN
        LOOP
        . ."PRIMES" ;
```

(SEE WHAT I MEAN ABOUT NOT BEING EASY TO READ!)

Listing 3

```
10 FORI=1TO1001:A%=2*2:NEXT I
```

Listing 4

```
:TEST 1000 0 DO 2 2 * DROP LOOP ;
```

..........

## WRITING & READING DISK FILES - PART 2

(Continued from Page 3)

size. Enough blanks are then put on the right-hand side to guarantee that the variable exceeds its defined size. Finally, the variable is truncated to the defined size.

To make a single variable, you can concatenate the input variables to, say X$, by writing -
X$=A$+B$+C$+D$+E$+F$+G$+H$+I$+J$

Let's ignore, for the moment, the problem of how to write and read two of these to each physical disk record. Just assume you wrote X$, as above, and then read it back.

A crude way to print the "label-only" data to the screen is to say -

```
  PRINT LEFT$(X$,28): !Prints Name
  PRINT MID$(X$,29,28): !Prints Address Line 1
  PRINT MID$(X$,53,28): !Prints Address Line 2
  PRINT MID$(X$,87,13)+",";MID$(X$,100,2)+" "+
        MID$(X$,102,9): !Prints City, St, & Zip.
```

I hope I haven't lost you. Perhaps you are wondering why one would want to go through all this, so let me review the situation.

(1) By limiting record size to 125, I can store two logical records per disk sector, thereby saving disk space.
(2) By forcing each field to a specified size, I am guaranteed that I won't overflow my 125 character record size.
(3) By forcing each field to a specified size, I know where each field starts and stops in the concatenated string. If I know this, then I don't need to use delimiters and therefore I can save, in this case, 10 characters which can be used for data.

The negative sides of the situation are:

(1) You can't input data of greater lenght than allowed in the definition. For instance, if I wanted the state to be "Ala.", I can't do it. If a city's name exceeds 13 characters, then that name must be abbreviated.
(2) You have to do your own parsing. That is, when you want to work on the phone field, you don't have it in I$. You can put it there (parse the input string X$) by saying -

```
     I$=MID$(X$,114,7)
```

To write to the disk, I use the following technique.

```
10 DIM X$(150),Z$(250)
20 I=1: ! "I" used for indicating the
     (1) left or (2) right
30 GOSUB 100: ! Get one logical record input
40 GOSUB 200: ! Move it to output buffer.
50 GOTO 30

100 INPUT A$,....,J$
190 X$=A$+B$+.....+J$
195 RETURN

200 IF I=1 THEN Z$=X$:I=I+1:RETURN
210 Z$=Z$+X$:PUT 1 Z$:I=1:RETURN
```

Of course there's a bit more to it than the above. I have to check for the operator ending input, for the end of input being in mid-physical record, or for the start of input being in mid-physical record.

The data is, as I've shown, written to the disk as a set of two 125-character strings concatenated into one. I write no delimiter to the disk. The delimiter, you remember, is used by the operating system to parse the separate strings out of a batch of data. Since I know where my strings start and stop, I need no delimiter.

Regardless of my needs, the operating system still looks for delimiters. As mentioned last month, the default delimiter is the comma (,). If I now write

```
     DIM Z$(250)
     GET 1 Z$
```

the operating system will physically read all of the sector into an internal buffer and then scan the buffer for a comma. If a comma exists, the operating system will put only the data preceeding the comma into Z$. That's not what I want, so I must change the string delimiter to be some character which will not by found in the data. A reasonable one to use is

```
     STRING CHAR$(255)
```

which is a HEX FF.

## Getting Data Back Out

Pictorally, if you input a series of logical records to a system such as we've discussed,the data is on the disk like this:

```
                    .........................
Physical Record 1   : Logical 1 : Logical 2 :
                    .........................
Physical Record 2   : Logical 3 : Logical 4 :
                    :........................
Physical Record 3   : Logical 5 : Logical 6 :
                    .........................
```

By inspection, you can see that if you wanted logical record 5, you'd have to get the left 125 characters of physical record 3. Stated mathematically:

```
     if P=Physical record, and
        L=Logical record
 then   P=INT((L+1)/2)
```

Working through a couple example to show you it works:

```
   for L=1    P=INT((L+1)/2)=INT((1+1)/2)=INT(1)=1
        =2                  =INT((2+1)/2)=INT(1.5)=1
         3                  =INT((3+1)/2)=INT(2)=2
```

Determining which side of the physical record can be done with the evaluation of the statement:

```
     MOD(L,2)
If MOD(L,2)=0, then use the right side.
If MOD(L,2)=1, then use the left side.
```

Next month I'll put this all together. Included will be how to pad when the operator stops in mid-sector and how to pick up the blank record on subsequent data entry.

..........

## RELOCATING MDOS HAS BENEFITS

by Ron Shenk
329 Robin Hood Rd., Atlanta, Ga. 30309

After adding a Micropolis Mod II disk drive to my SOL computer I had two BASIC intrepreters, PT's ECBASIC, which I had been using up to that time, and the Micropolis disk basic, which I call MBASIC. Both have nice features: MBASIC supports random access disk files, can achieve up to 60 decimal digits of accuracy if desired, and has nice program tracing. On the other hand, ECBASIC supports IF..THEN..ELSE, has a gem of an editor and, most significantly, is FAST. Typically, MBASIC requires 30% more time to execute the same program.

How nice it would be if ECBASIC could be interfaced to the disk drive.

The solution I reached was to relocated MDOS to the top of RAM memory and write a disk interface routine, which I call INTECOM. With this relocated operating system, RDOS, several potentially useful applications became possible. I could now use DEBUG to trace programs which occupy low memory overwriting MDOS. I could even write off to disk the disassembled listing produced by DEBUG. I could also write to disk listings made by LINEEDIT of MBASIC in order to use features offered by a second editor or to execute MBASIC programs by ECBASIC. I could download to disk BASIC programs listed from a time sharing service or other remote site and then feed them to MBASIC for execution without having to type a single line, if the syntax was compatible. I could convert CP/M files to MDOS files and vice-versa.

To relocate MDOS, I first found all addresses in MDOS (about 1600 in number). Then I wrote the program MDOSMOVER to increment each one of them by XX00H, for any hex digits XX, and move the resulting code to the new location. MDOSMOVER works with version 4.0 of MDOS and with my particular I/O routines. However it should not be difficult to modify by hand the I/O section of MDOS for any particular computer.

For anyone interested, I can supply the programs MDOSMOVER or INTECOM on printout, on Micropolis Mod II diskette, or on SOL-CUTS cassette tape. For printout, please send $3 for either program or $4 for both. For diskette or tape, please send $15 for either program, or $24 for both.

..........

## SORCERER'S APPRENTICE

The Sorcerer is alive and well and living in Michigan.

The Sorcerer's Apprentice Users Group publishes a fact-filled newsletter eight times per year. Content of the newsletter is directed at the Sorcerer user, from the true apprentice to the experienced business user with a disk system.

Circulation is world-wide, and articles in the newsletter are contributed from individuals and from editors who address a specific subject each issue. The Sorcerer's Apprentice is in its fourth year of publication.

Current issues of the Sorcerer's Apprentice are averaging 24 pages, magazine-size, with both articles and advertising of interest to the Sorcerer user. Back issues are available.

The users group also maintains a Sorcerer CBBS system in the Detroit area (area 313, 535-9186 ringback).

Sorcerer users around the world are invited to join the Sorcerer's Apprentice Users Group. Annual dues are $18 in the United States, $24 Canada and Mexico, and $32 in other countries. Dues include the newsletter, access to the Sorcerer CBBS, and other services.

For information, write Sorcerer's Apprentice, P.O. Box 33, Madison Heights, Michigan, 48071. Or, call president Don Gottwald, (313) 286-9265.

..........

## LIBRARY CHANGES & ADDITIONS

As I mentioned last month, I'm trying to get the library disks to contain 35 tracks or less, and to get them organized into categories.

Disk 6 now contains only System Utilites. These work, I think, and in many cases contain the source code, so you can modify them. The MDOS dissassembly of Manderson's has been moved to Disk 17. The System Patches have been moved to Disk 16.

I moved the MUG Newsletter Indexing programs and data files from Disk 7 to Disk 13. Actually, these are all new. Ken Findlay rewrote the programs and files to pack 3 logical records to each sector. Thanks alot, Ken. We were running out of space with the old format. And I know that conversion wasn't a trivial task. The Newsletter index for issues 13-20 is printed on pages 13-16. The non-interfaced menus on Disk 7 were deleted.

Disk 9 now also contains only System Utilities. The system patches were moved to Disk 16, and the FILECOPY disassembly moved to Disk 18; System Disassemblies And Documentation. Manderson's Z80 disassembler was moved to Disk 15.

There is some new material on the above mentioned disks, but I'll discuss it next month.

Totally new material exists on the following disks: Disk 12 - Technical & Household programs; Disk 14 has some miscellaneous routines which aren't categorized; Disk 18 has some further info & documentation for Manderson's DMOS disassembly (Disk 17); Disk 19 has some new games, and Disk 20 has FORTH.

There probably is enough for CP/M disks 1003 (Games) & 1004 (Utilities), but I can't be specific about their contents at the time I'm writing this.

I really hope that this shifting around doesn't upset you terribly. In the long run, I'm quite sure it's a proper move, but it will cause some confusion for a time.

To brighten things up, remember that all disks from 6 on out are now $10 instead of $15 ($12 instead of $17 for those of you outside of North America). For those of you submitting programs on your disk, the price remains $3 ($5 outside North America). MOD I owners no longer have to send a second disk.

..........

## LATEST DISK CONTENTS

What follows are updated Table of Contents for some of the Library Disks. Space doesn't permit all updates to be listed this month, so I'll finish next issue.

MUG MDOS Library Disk 06, Revision 06, APR 82
SYSTEM UTILITIES - 25 tracks

| Name | Typ | Rv | Sze | Author/Description |
|---|---|---|---|---|
| EDITTEXT | SRC | 00 | 013 | Manderson, R. - Documentation for the following 9 files. |
| LINEEDIT5 | SYS | 02 | 00F | Manderson, R. - The assembled version of the EDIT files. Similar to Mp LINEEDIT, but has MERGE, enhansed APPEND, & editing capabilities. |
| SYSQ1 | SRC | 00 | 00B | Manderson, R. - |
| EDIT1 | SRC | 00 | 00B | Manderson, R. - |
| EDIT2 | SRC | 00 | 021 | Manderson, R. - |
| EDIT3 | SRC | 00 | 027 | Manderson, R. - |
| EDIT4 | SRC | 00 | 016 | Manderson, R. - |
| EDIT5 | SRC | 00 | 01B | Manderson, R. - |
| EDIT6 | SRC | 00 | 013 | Manderson, R. - |
| DISYMS | SRC | 00 | 021 | Manderson, R. - Instructions for assembly & use, as well as the source for a Z80 disassembler. |
| UNASSM | SYS | 00 | 00B | Manderson, R. - The assembled version of DISYMS. Lots of options - lists, ASCII equivalents, SYM tables. |
| S/DISKCOPY | SRC | 00 | 014 | Findlay, K. - Source for the following substitute for the Mp DISKCOPY. Slower but surer on some systems. |
| DISKCOPY | SYS | 00 | 004 | Findlay, K. - |

........

MUG MDOS Library Disk 07, Revision 02, APR 82
MISCELLANEOUS - 32 tracks

| Name | Typ | Rv | Sze | Author/Description |
|---|---|---|---|---|
| INSTR.TEXT | TXT | 00 | 013 | Shapiro, J. - Documentation for the two INSTR. programs that follow. |
| INSTR.GEN | BAS | 00 | 00E | Shapiro, J. - Allows generation/ read of data files that contain a program's instructions & documentation. Frees memory & speeds up programs. See newsletter #15. |
| INSTR.READ | BAS | 00 | 00B | Shapiro, J. - |
| TOKEN | BAS | 00 | 007 | Powers, W. - Hunts down the tokens, or codes, used in Mp BASIC. Lists codes & function. See newsletter #14. |
| DISKMENU | BAS | 00 | 007 | Burkhardt, E. - Reads the dir- ectory (DIR) file from video mapped memory. Outputs to screen. See news'r #12. |
| LABELS | BAS | 00 | 007 | Burkhardt, E. - Same as above, but prints disk labels. |
| W-2S.DOC | TXT | 00 | 004 | Schoenke, T. - Documentation for W-2S. |
| W-2S | BAS | 00 | 00F | Schoenke, T. - Produces W2s & W3s for those whose payroll is manual or whose program doesn't do it. |
| JUMBLES | BAS | 00 | 008 | Riding, G. - Solves the JUMBLES puzzle contained in many newspapers. See news'r #16. |
| TAXDEPR | BAS | 00 | 011 | Rothstein, M. - Generates depreciation schedules for S/L, 125%DB, 150%DB, DDB or Variable method. Also Investment Credit & Bonus depreciation. |
| CW.DOC | TXT | 00 | 00B | Fait, D. - Documentation for the following 8080 assembly language program. |
| CW.SRC | SRC | 00 | 02D | Fait, D. - Converts keyboard input to a CW sequence, via a look-up table, & sends to output port. Receives & times signal from input port, con- verts to char. via look-up table, & displays it. |
| CW | OBJ | 00 | 006 | Fait, D. - |

PER-REM BAS 00 00C Ivey, W. PERSPECT. documen'n.

PERSPECT. BAS 01 022 Ivey, W. Aids artist or draftsman in producing perspective drawings.

TEST1-I DAT 00 00F Ivey, W. -

TEST1-O DAT 00 00F Ivey, W. -

CHECKBOOK BAS 00 00C Little, H. - Checkbook balanc'g.

MOTION BAS 00 005 Pickert, A. - Illustration of screen motion. See news'r #16.

RESCUE.DOC SRC 00 016 Bohn, J. - RESCUE documentation

RESCUE OVL 00 004 Bohn, J. - Allows unSCRATCHing of files. Handy utility.

SPELL BAS 00 003 Harden, Jon - Neat tool for learning. Takes list, displays word for variable time. Then clears screen. You must now type the word correctly.

INSERT BAS 00 003 Harden, Jim - Binary insertion subroutine.

STOCK-MRKT BAS 00 00D Harden, Jon - Tool for personal stock decisions.

........

MUG MDOS Library Disk 09, Revision 02, APR 82
SYSTEM UTILITIES - 22 tracks

Name Typ Rv Sze Author/Description

==== === == === =====================

$BATCHCOPY SRC 01 01B Singer, C. - Source for a batch FILECOPY. Can move up to 50 files in a single call. Doesn't have to be resident on any disk involved in the copying.

BATCHCOPY USR 01 005 Singer, C. - Executable version.

$BLINKCOPY SRC 00 022 Singer, C. - Source for a modified BATCHCOPY that moves cursor as files are copied.

$SEARCH SRC 00 00C Singer, C. - Replacement for SEAR to allow string searches.

#RESTORE SRC 00 00C Singer, C. - Finds and reconstructs SCRATCHed files.

RESTORE USR 00 006 Singer, C.

#GET-TRAX SRC 00 019 Singer, C. - General purpose utility for inspection of disk tracks without recourse to DIR.

GET-TRAX USR 00 005 Singer, C.

$SYMSHELL SRC 00 01A Singer, C. - Symbol Sort-List Utility.

SYMSHELL USR 00 004 Singer, C.

$RENUM SRC 00 049 Singer, C. - Utility for renumbering BASIC lines for Version 3.0.

RENUM SYS 00 008 Singer, C.

..........

MUG MDOS Library Disk 11, Revision 01, APR 82
SYSTEM UNIQUE - 35 tracks

These programs contain screen control codes for specific processors, and must, in most cases, be be modified before use in your system.

NAME TYP RV SZE AUTHOR/DESCRIPTION

==== === == === =====================

DOCGEN.BAS BAS 00 032 Jamba, D. This, and the following ten files, are a document generator (word processor), written in BASIC. Contains EXIDY dependent code and must be modified for operation on other systems.

VUTIL..OBJ OBJ 00 002 Jamba, D.

DGCONT.BAS BAS 00 008 Jamba, D.

DGENTR.BAS BAS 00 02E Jamba, D.

DGLIST.BAS BAS 00 00F Jamba, D.

DGSAVE.BAS BAS 00 007 Jamba, D.

DGREAD.BAS BAS 00 006 Jamba, D.

DGEROR.BAS BAS 00 004 Jamba, D.

DGALTR.BAS BAS 00 00F Jamba, D.

DOCGEN.LIB DAT 00 013 Jamba, D.

DOCGEN.DOC BAS 00 019 Jamba, D.

SINITIAL SRC 00 064 Hall, L. Converts North Star disks to Micropolis - if you have BOTH controllers in your system.

LIFE BAS 00 008 McGraw, D. This, and the following eight files, are the game of LIFE. Contains Compucolor dependent code and must be modified for operation on other terminals.

SRCDIS SRC 00 003 McGraw, D.

SCRSET SCR 00 010 McGraw, D.

SRCCOM SRC 00 00B McGraw, D.

SETUP OBJ 00 003 McGraw, D.

DISPLAY OBJ 00 002 McGraw, D.

COMPUTE OBJ 00 002 McGraw, D.

INSTRUCT BAS 00 00F McGraw, D.

DEARBUZZ SRC 00 00C McGraw, D.

MUG MDOS Library Disk 12, Revision 00, APR 82
TECHNICAL & HOUSEHOLD - 27 tracks

Name Typ Rv Sze Author/Description

==== === == === =====================

MENU BAS 00 00C Rudow, B. - Master Menu, including auto-configuration.

MENU.T BAS 00 007 Rudow, B. - Submenu for technical programs.

MENU.H BAS 00 007 Rudow, B. - Submenu for household programs.

DATE.CK BAS 00 006 Rudow, B. - Verify/Change System Date.

FLIGHTPLAN BAS 00 00B Ripley, B. - Plans aircraft flight plans, including time and fuel.

EARTHSTAT BAS 00 007 Helm, E. - Computes azimuth and elevation look angles for geosyncronous satellites.

VOR/DME-DR BAS 00 043 Lugar, J. - A program for aircraft navigation using only the standard Vortacs to provide Great Circle Direct navigation that is normally only available to aircraft equipped with expensive area navigation receivers.

TRIANGLE BAS 00 014 Lugar, J. - This program is an engineering program that computes with trignometry the remaining parts of any triangle given just three parts.

AREA BAS 00 00B Lugar, J. - This program computes the area of a figure or a parcel of land that has straight sides or a circular curved side between known coordinate points. This is useful to engineers, architects, and builders.

INVERSE BAS 00 009 Lugar, J. - This program computes the bearing and distance between any known coordinate points. This also uses Coordinate Geometry and is most useful to those involved in land parcels and building.

CURVE BAS 00 024 Lugar, J. - This is an engineering program to compute the standard inter-related parts of any circular curve knowing any two parts. Also part of the program gives offset dimensions to aid in staking or plotting any curcular curve.

TOURISTCRD BAS 00 009 Rusczyk, R. - Prints a formletter postcard requesting tourist information.

POLCARD BAS 00 010 Rusczyk, R. - Prints a formletter postcard to your politicians.

BUYHOUSE BAS 00 017 Rusczyk, R. - Computes and prints an 3-page evaluation which is useful in selecting a used house for personal or investment purposes.

STOCKCARD BAS 00 009 Rusczyk, R. - Prints a postcard request for quarterly and annual corporation report.

HOUSEREPAR BAS 00 00E Rusczyk, R. - Maintains a record of all the repairs and enhancements to your home and property.

HOUSELIST DAT 00 007 Rusczyk, R. - Data file for HOUSEREPAR.

OILTANK BAS 00 00C Rice, J. - Calculates the gallon capacity for 1/2 inch increments of any cyclindrical tank.

..........

MUG MDOS Library Disk 18, Revision 00, APR 82
SYSTEM DISASSEMBLIES AND DOCUMENTATION - 7 tracks

Name Typ Rv Sze Author/Description

==== === == === =====================

#MDOSDOC1 DOC 00 009 - Rusczyk, R. - Questions to Manderson about separating RES & MDOS.

#MDOSDOC2 DOC 00 02A Manderson, R. - Answers.

#MDOSDOC3 DOC 00 006 Rusczyk, R. - Further problems and comments.

$FILECOPY SRC 00 00E Singer, C. - FILECOPY disassembly.

..........

MICROPOLIS USER'S GROUP NEWSLETTER INDEX VOLUMES 13 - 20 / ALPHA BY TOPIC

| PGMNAME/TOPIC | CO./AUTHOR | PGM/ARTICLE TYPE | CATEGORY | VOL-PG |
|---|---|---|---|---|
| A-FORTH COMPILER | ACROPOLIS | HIGH LEVEL LANG | COMPILER | 020-02 |
| ACCESSING DISK FILES* | B.RUDOW | BASIC PGM TECHNIQUE | DISK FILES | 020-03 |
| AUTO-CONFIGURATION* | B.RUDOW | BASIC PGM TECHNIQUE | TERM I/O | 019-08 |
| BASIC PGMING | | BASIC PGMING | | 013-01 |
| BASIC PGMING | | BASIC PGMING | | 015-01 |
| BASIC PGMING | | BASIC PGMING | | 016-01 |
| BASIC PGMING* | B.SMITH | BASIC PGMING | | 014-01 |
| BASIC TOKENS | | BASIC STATEMENTS | REFERENCE | 018-02 |
| BASIC TOKENS* | W.POWERS | BASIC DOC | REFERENCE | 014-05 |
| BASIC TOKENS/KEYWORDS | | BASIC DOC | REFERENCE | 020-13 |
| BASIC/S + BASIC/Z | SYSTEMATION | HIGH LEVEL LANGUAGE | COMPILER | 020-06 |
| BASIC/S COMPILER | SYSTEMATION | BASIC SYSTEM PGM | COMPILER | 013-01 |
| BASIC/S COMPILER* | SYSTEMATION | BASIC SYSTEM PGM | COMPILER | 014-01 |
| BATCHCOPY* | C.SINGER | 8080 UTILITY PGM | DISK FILES | 017-06 |
| BOOKKEEPING | DATASMITH | BASIC APPL PGM | BUSINESS | 020-09 |
| CDS VERSATILE COMPUTER | CDS | HARDWARE | | 014-02 |
| CLASSIFIED ADS | | GENERAL INFO | | 013-06 |
| CLASSIFIED ADS | | GENERAL INFO | | 014-08 |
| CLASSIFIED ADS | | GENERAL INFO | | 015-08 |
| CLASSIFIED ADS | | GENERAL INFO | | 016-16 |
| CLASSIFIED ADS | | GENERAL INFO | | 017-16 |
| CLASSIFIED ADS | | GENERAL INFO | | 018-16 |
| CLASSIFIED ADS | | GENERAL INFO | | 019-12 |
| CLASSIFIED ADS | | GENERAL INFO | | 020-14 |
| COMMUNICATIONS | | 8080 SYSTEM PGM | | 018-03 |
| COMPAL-8200 | | HARDWARE | | 016-11 |
| COMPUTER NO'S | | PGMING THEORY | | 017-01 |
| CONTROL-P | | 8080 SYSTEM PGM | OP SYSTEM | 018-08 |
| CONTROL-P* | B.CARIGNAN | 8080 SYSTEM PGM | OP SYSTEM | 018-12 |
| CP/M | | 8080 SYSTEM PGM | OP SYSTEM | 013-02 |
| CP/M | | 8080 SYSTEM PGM | OP SYSTEM | 014-02 |
| CP/M - MDOS COMPARISON | | 8080 SYSTEM PGM | OP SYSTEM | 014-06 |
| CP/M - MDOS CONVERSION* | B.RUDOW | UTILITY PGM | | 019-06 |
| CP/M HANG FIXES | | 8080 SYSTEM PGM | OP SYSTEM | 020-12 |
| CURSOR CONTROLS* | J.FACTOR | BASIC PGM TECHNIQUE | TERM I/O | 013-03 |
| DEBUG | | 8080 SYSTEM PGM | DEBUGGER | 017-04 |
| DIM | | BASIC STATEMENT | REFERENCE | 019-01 |
| DISK CATALOG SYSTEM* | B.RUDOW | BASIC UTILITY PGM | DISK DIR | 017-10 |
| DISK DRIVE BELTS | | HARDWARE | | 016-13 |
| DISK ERRORS | | DISK MEDIA | | 014-06 |
| DOCGEN | | BASIC APPL PGM | WORD PROC | 018-09 |
| DOUBLING KEYS | | 8080 SYSTEM PGM | OP SYSTEM | 018-08 |
| DRIVE TURN-OFF | | HARDWARE | | 016-03 |
| FORTH | | HIGH LEVEL LANGUAGE | COMPILER | 013-02 |
| FORTH | | HIGH LEVEL LANGUAGE | COMPILER | 013-03 |
| GENERAL LEDGER PGMS | MODERN MICRO | BASIC APPL PGM | BUSINESS | 013-03 |
| GET-TRAX | | 8080 UTILITY PGM | DISK FILES | 018-06 |
| GRAPHICS* | A.PICKERT | BASIC PGM TECHNIQUE | | 016-02 |
| IBM/MICROPOLIS FORMATS | | HARDWARE | | 020-12 |
| INTERNAL DATA FORMAT | | BASIC DOC | REFERENCE | 020-01 |
| INTERRUPTS | | HARDWARE | | 015-05 |
| JUMBLE PUZZLES* | G.RIDING | BASIC APPL PGM | GAME | 016-04 |
| LATAH | | HARDWARE | | 017-02 |
| LIFE | | BASIC APPL PGM | GAME | 018-05 |
| MDOS ALTERATIONS | | 8080 SYSTEM PGM | OP SYSTEM | 014-04 |
| MDOS HANG FIXES | | 8080 SYSTEM PGM | OP SYSTEM | 019-05 |
| MDOS HANG FIXES | | 8080 SYSTEM PGM | OP SYSTEM | 020-12 |
| MDOS PDS VERSION 4.0 | MICROPOLIS | 8080 SYSTEM PGM | OP SYSTEM | 014-03 |
| MDOSPATCH | | 8080 SYSTEM PGM | | 018-09 |
| MERGE | DATASMITH | BASIC PGMING AID | | 020-09 |
| MICRO-LINK | | 8080 SYSTEM PGM | | 016-14 |
| MICROPOLIS HARDWARE | MICROPOLIS | HARDWARE | | 013-02 |

MICROPOLIS USER'S GROUP NEWSLETTER INDEX VOLUMES 13 - 20 / ALPHA BY TOPIC

| PGMNAME/TOPIC | CO./AUTHOR | PGM/ARTICLE TYPE | CATEGORY | VOL-PG |
|---|---|---|---|---|
| MICROPOLIS HARDWARE | MICROPOLIS | HARDWARE | | 013-05 |
| MICROPOLIS PDS SUMMARY | MICROPOLIS | 8080 SYSTEM PGM | OP SYSTEM | 019-04 |
| MICROPOLIS PREV MAINT | | HARDWARE | | 018-04 |
| MICROPOLIS S/W | | GENERAL INFO | REFERENCE | 018-01 |
| MICROPOLIS SOFTWARE | | REFERENCE | LISTING | 020-14 |
| MODI-MODII CONVERSION | | HARDWARE | | 014-05 |
| MODI-MODII CONVERSION | | HARDWARE | | 014-06 |
| MODII-MODI CONVERSION | | HARDWARE | | 013-02 |
| MOTION* | A.PICKERT | BASIC PGM TECHNIQUE | | 016-02 |
| MUG CONTROL OF MDOS | | GENERAL INFO | | 018-03 |
| MUG DISK MASTER MENU* | B.RUDOW | BASIC PGM TECHNIQUE | TERM I/O | 019-12 |
| MUG LIBRARY | | DISK06 CONTENTS | REFERENCE | 014-04 |
| MUG LIBRARY | | GENERAL INFO | PRICE LIST | 016-05 |
| MUG LIBRARY | | DISK01 DIR LISTING | REFERENCE | 016-06 |
| MUG LIBRARY | | DISK02 DIR LISTING | REFERENCE | 016-07 |
| MUG LIBRARY | | DISK03 DIR LISTING | REFERENCE | 016-08 |
| MUG LIBRARY | | DISK04 DIR LISTING | REFERENCE | 016-08 |
| MUG LIBRARY | | DISK06 DIR LISTING | REFERENCE | 016-09 |
| MUG LIBRARY | | DISK05 DIR LISTING | REFERENCE | 016-09 |
| MUG LIBRARY | | DISK07 DIRT LISTING | REFERENCE | 016-10 |
| MUG LIBRARY | | DISK06 MODIFICATIONS | REFERENCE | 016-12 |
| MUG LIBRARY | | DISK09 DIR LISTING | REFERENCE | 018-11 |
| MUG LIBRARY | | DISK08 DIR LISTING | REFERENCE | 018-11 |
| MUG LIBRARY | | DISK10 DIR LISTING | REFERENCE | 018-12 |
| MUG LIBRARY | | DISK11 DIR LISTING | REFERENCE | 018-12 |
| MUG LIBRARY | | DISK1001 CONTENTS | REFERENCE | 020-10 |
| MUG LIBRARY - STDS | | GENERAL INFO | | 020-11 |
| MUG LIBRARY RULES | | GENERAL INFO | | 020-14 |
| MUG NEWSLETTER INDEX | | GENERAL INFO | REFERENCE | 015-05 |
| MUG OBJECTIVES | | GENERAL INFO | | 014-07 |
| MUG SOFTWARE HOUSE | | GENERAL INFO | | 019-01 |
| PASSWORDS | | BASIC PGM TECHNIQUE | | 018-08 |
| PENSION PGMS | | BASIC APPL PGM | BUSINESS | 014-06 |
| PGM DOCUMENTATION* | J.SHAPIRO | BASIC PGMING | | 015-03 |
| PONY-PICK | BONJOEL | 8080 APPL PGM | | 019-08 |
| PUBLISHED SOFTWARE | | GENERAL INFO | | 013-04 |
| READING DISK DIR | B.ZALE | BASIC UTILITY PGM | DISK DIR | 014-04 |
| RESTORE FILES | | 8080 UTILITY PGM | DISK FILES | 018-05 |
| RIGID DISKS | | HARDWARE | | 018-03 |
| SAVING BASIC PGMS | | BASIC PGM TECHNIQUE | | 018-15 |
| SCREEN DUMP | | 8080 SYSTEM PGM | OP SYSTEM | 018-08 |
| SCREEN EDITOR | | 8080 UTILITY PGM | EDITOR | 018-08 |
| SEARCH | | 8080 SYSTEM PGM | | 018-09 |
| SGN FUNCTION | | BASIC PGM TECHNIQUE | | 014-05 |
| SIZE(N)* | G.RIDING | BASIC PGM TECHNIQUE | DISK FILES | 016-02 |
| SMASH | DATASMITH | BASIC PGMING AID | | 020-09 |
| SOL 80-COLUMN DISPLAY | | HARDWARE | | 018-14 |
| SORCERER BASIC | | HIGH LEVEL LANG | INTERPRET | 018-13 |
| SORCERER S/W | | UTILITY PGMAS | | 016-15 |
| SORCERER/MDOS PATCHES | | HARDWARE | | 019-12 |
| STATISTICS PGMS | MODERN MICRO | BASIC APPL PGM | BUSINESS | 013-04 |
| STRING ARRAYS* | B.RUDOW | BASIC PGM TECHNIQUE | STRINGS | 019-02 |
| STRING OPERATIONS* | K.FINDLAY | BASIC PGM TECHNIQUE | STRINGS | 016-13 |
| SYSTEM MEMORY | | 8080 PGM TECHNIQUE | | 017-03 |
| TABLE HANDLING | | BASIC PGM TECHNIQUE | | 014-02 |
| TABLE HANDLING* | J.HARDEN | BASIC PGM TECHNIQUE | | 014-07 |
| TABLE HANDLING* | N.DEMBINSKI | BASIC PGM TECHNIQUE | | 015-01 |
| TAXPRO | | 8080 APPL PGM | BUSINESS | 017-04 |
| TEXTWRITER | | 8080 APPL PGM | WORD PROC | 017-05 |

MICROPOLIS USER'S GROUP NEWSLETTER INDEX VOLUMES 13 - 20 / ALPHA BY TOPIC

| PGMNAME/TOPIC | CO./AUTHOR | PGM/ARTICLE TYPE | CATEGORY | VOL-PG |
|---|---|---|---|---|
| TRACK DENSITIES | | 8080 SYSTEM PGM | OP SYSTEM | 017-04 |
| VERSATILE CDS | | HARDWARE | | 017-14 |
| Z80 ASSEMBLER | | 8080 PGMING AID | ASSEMBLER | 018-10 |

## CLASSIFIED

FOR SALE: UDS-103A-CBS, Bell-103 compatible 300 baud data modem. Answer only, direct connect, with a direct access arrangement (DAA). $100.

Joe Callaway, 1728 51st St. West
Birmingham AL 35208 - (205) 925-8169

..........

FOR SALE: S100 system with 48K RAM. North Star Z80A processor. Dual Micropolis MOD II drives. Visual 200 display terminal. Decision Data 6540-1 printer. Micropolis software only. Complete for $3,950. Will sell printer only for $850. Call Call (313) 887-8136 and ask for Ray.

..........

WANTED: Help in interfacing a VG Flashwriter II board to a TEI Video Controller Card MCS-VCB. Running with a TEI CPU.

Bob Samwel, AB Datatel Inc., Box 30, S-293 01 Olofst Sweden. Phone +46-45441325, TELEX: 4545 DATATEL S.

..........

WANTED: Help in patching CP/M 1.42 on a Vector Graphic. Need the code for CBIOS and the code for the VG monitor which drives a 48K terminal I/O. Contact Mark Levy, 2825 E. Hillery, Phonix AZ 85032. Phone (602) 867-2101 (home) or 937-1666 (work).

..........

## DAMAN SOFTWARE

The DAMAN (software sales) side of the MUG is surviving, though the purchase of inventory is expensive. I sincerely appreciate the business of you MUG members. Hopefully, everyone is pleased with the software. I'm slowly learning the capabilities of the S/W we're offering. If there are questions on your particular needs, please call or write.

One program I thought would sell better than it has, was Spellbinder.

I have, available for immediate delivery, one copy of Spellbinder for each of the following configurations.

(1) Vector Graphics (56K, 4.0 monitor or greater)
(2) Menu selectable I/O terminal - includes Hazeltine 1500, ADM 3A, Intertube II, Heath-19, SOROC-120, TVI-910; 912; 920; 925; 950, ZENTEC, Cromemco 3102, Visual 200, and ADDS Viewpoint. Any terminal or video mapped memory system can be configured, but it takes modification of an assembly language file, and a DDT job to do it.

Spellbinder is a full function wordprocessor with macro capability which allows features to be added for the unique requirements of each user. A Mail List macro is included for mail merge with form letters. Spellbiner incorporates a full screen video editor, drivers for regular or precision printers, has soft hypenation, insertion/deletion, search (and replace/delete/add/with wildcards), a print to screen to preview text as it will appear on the page, and all the printing enhansements and controls.

DAMAN's price for Spellbinder is $319 (lists at $495), delivered (add $7 outside North America). There is a version for the Exidy, though I don't have it in stock.

..........

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
===== ===== ====