MICROPOLIS USERS GROUP

MUG Newsletter #23 - June 1982

****

NOTES FROM THE EDITOR

RENEWAL TIME

Here we are, approaching the end of another MUG year. Most of you are on the Aug. to July year and need to send in next year's dues. I thought about raising the dues $2, but decided against it. So, in North America, it's still $18. Elsewhere it's $25. In either case, you can use VISA or Master Card with no penalty. (I gave up the 5% surcharge also.) I hope you have received enough benefit to justify your renewal. The North American charge for the year-one back issues will go from $12 to $18, though. Outside North America, back issues remain at $25.

MUG DIRECTION

But where are we going? I'd really appreciate you sending a note with your renewal that tells me what direction we should go. Historically, we started out looking at Micropolis Basic. Discussion of available commerical MDOS software was always an aim, of course. Then the disassembleis of MDOS and the other utilities broadened the scope. Soon we were discussing some pretty deep systems programming in assembler language. Some members (including myself, most of the time) find these topics over their head.)

The next evolution was CP/M. This is a broad subject - including many variations of that operating system (CP/M, itself), several Basics, and loads of other languages, utilities and application programs. Some members have zero interest in CP/M, some run it exclusively. CP/M is covered in all the slick magazines. The question is, what is the MUG's contribution?

Other extensions of coverage are caused by DAMAN's software reviews/promotions, and BASIC/S-Z. I personally believe that the DAMAN side fits, since I'm discussing available software, and offering it at a discount. Perhaps you don't see it that way, however. I think BASIC/S-Z is terrific and I'd like to share my experiences, and those of other member/owners, with the group. The majority of you don't own either version, though, and that coverage is therefore useless to you.

Things I'd like to know about are numerous. Whether you'd like to know is the question. Do we want hardware reviews on S-100 products you might use? Products include voice and sound boards, printer buffers, disk-on-a memory-board, graphics, color, communications, multiuser systems, networking in general, hard disks, printers, terminals, general A/D-D/A (interface to the external world) - the list is large.

So, the question to you fellows and gals is - "What?". What is the purpose of the MUG these days? Should there be sub-sets of membership? Maybe there should be several smaller $12 newsletters for MDOS, CP/M and BASIC/S-Z. I don't know. I've just been moving along, publishing what has been comming in from you members ans what has been of interest to me. So let me know. Even if you don't renew, let me know what it was you thought we were going to do that we didn't.

YOUR HELP IS APPRECIATED

One final thought. Remember, this is a users group newsletter, not a slick magazine. I appreciate the questions, articles and programs that have come in over the two years. That's what keeps us going. There's no way I can write all this stuff every month. If you're interested in a subject, write an

(Continued in column 13)

BUILDING THE CHEAP COMPUTER, PART II

by Zot Trebor

The title of this column of helpful hints for the digitally handicapped has drawn fire from the very group it was intended to help. Bitter pill. It seems that a few people have wondered how I could have the gall to use the expression 'cheap computer' in conjunction with Micropolis disk drives. The answer is simple; I inherited the disk drives. My mother had ten brothers and one of them left me a Micropolis Mod 1053-II, a collection of Victorian pornography, and the fishing rights to a river in Iceland.

And with that as my final comment on the subject of cheap Micropolis drives, let's return to the Cheap Computer.

Actually, the SSM boards, although chosen for their price, are an excellent value. I had rather hoped to own a Cromemco S-100 cpu board. Those little rascels have EVERYTHING on one board; serial I/O, RAM, ROM, a wet-bar and several functions I couldn't pronounce. They were also expensive. The SSM cpu on the other hand has only RAM and ROM. No wet-bar. No seial I/O. No mortgage requirements to own one either. I bought the cpu on sale from JADE. I'm not sure it was a good idea. JADE had obviously bought the boards from SSM; at least I hadn't heard of any highjacking lately. JADE then used its staff of highly skilled engineers and technicians to carefully select the necessary chips, sockets and hardware to make up the complete kit. This might not be the best way to go. One of the 8212 latches was bad and several of the chips were so old the countries they had been made in no longer existed. I mean OLD. One of the 7400's still required filiment voltage... But the documentation was at least up to the usual hobbiest standards for kits. One page said, in impressively large letters, "Put the mother together!" The other page of the documentation said to plug it in.

There is an array of switches on the cpu that lets the builder set various options and addresses. One sets the keyboard port and status port. One switch sets the Jump-to address (the board has vector jump on reset) which also becomes the starting address of the on-board ROM's. There is also a very tiny little switch that lets you shut off the ROM's if you don't like them for some reason. You can also shut off the vector jump. Or turn it on. I've tried it both ways and the board still remains firmly in its socket so it's probably not developing full power on the jump.

There is a bit of RAM on the board; 256 words as I recall. A third switch lets you set its address and turn it on or off. It is possible to write very tiny programs in this space, such as jumping to itself, trajectory correction routines or two element look-up tables. I tucked it away up in F-country and left it turned on. One day I'll find it there and wonder what the hell it is.

The SSM video board is a stunner. It is the usual memory-mapped variety with sixteen lines of 64 characters per line. It has its own crystal and the character set is adequate for students of Greek, most mathematicians and players of dull games. The Roman characters have tails where needed. All in all it is a bargan priced piece of goods that has no right to work as well as it does. The board is very well laid out and the documentation, while not a tome, is adequate for all but the compulsive circuit-follower.

The really sad point of the system is the wreched monitor program SSM sells with it. They have taken two perfectly good 2708's and loaded them with the most unbelievable collection of poor programming. The poor things set there and actually pant when you try to get something on the screen. They are loaded with unused and largely unusable routines for paper tape punching and teletype printing and

(Continued in column 12)

MDOS UTILITIES

Exactly where the line is drawn between application programs and utility programs is a bit vague to me. I suppose it has to do with the distinction of producing a "deliverable" output. Utilities don't produce a product such as a letter, invoice, or set of labels. Utilities do aid the programmer in his use of the operating system, by allowing manipulation of memory, disk tracks and disk files.

Previous issues of the newsletter have lightly discussed the utilities of System/z and DATASMITH, vendors who have been part of the MUG since day one. This month I've included an in-depth look at the utilities of two vendors who have only recently come to our attention, Acropolis and GMS. Next month I'll cover the utilities of System/z and DATASMITH to an equal depth. If I'm real productive, I'll create a cross-reference list for the functions performed.

....

ACROPOLIS UTILITIES

The Acropolis utility programs have minimum system requirements (unless specified otherwise in a given program) of an 8080 or Z80 microprocessor with a minimum of 16K of memory and MDOS 4.0 (3.0 versions available upon request).

The programs are loaded and executed using the procedures listed in section 4.1.2 of the Micropolis MDOS manual. That is, any of the programs may be executed by typing the unit number which contains the utility diskette, followed by a colon, followed by the name of the program, followed by the parameters required by the program, i.e., input files, output files, disk drive numbers, etc. All of the programs follow this command procedure.

MDOS is called for all input parsing and disk access. Errors are handled with the familiar MDOS methods and messages. As in MDOS, all programs may be paused by typing control S, or cancelled by typing control C, at virtually any time during their execution. The programs are written to work well with either a 64 or 80 character wide terminal. No modifications should be necessary for either memory mapped (usually 64 characters) or serial (usually 80 characters) teminals.

The assembly source files on the disk are named the same as the assembly object programs except that they are followed by the string "S" to denote source listings. The instruction manual contains listings of all of the programs. You can modify the programs as you need. The programs are already assembled, however, so those of you who stay away from assembly language can continue to do so. For those interested, the education gained in seeing how George Shaw uses the callable MDOS subroutines is invaluable. I imagne you'll be able to use these programs as skeletons for developing utility routines of your own.

The Acropolis MDOS utility set is available from the MUG (DAMAN) for $45, postpaid. Add $7 outside North America. VISA and Master Card accepted. Please specify MOD I or II, MDOS Version 3 or 4. Included in the Acropolis MDOS utilities are VIEW, MTEST, ASCII MEMORY DUMP, RECOVER, VERIFY, COMPRESS, and TOKENIZE. Each is discussed below.

VIEW

VIEW is probably the most convenient program in the utility package. The program allows the user to view a LINEEDIT, text, data, or BASIC program file from MDOS. The user may specify that the output alternatively be directed to the printer with a title, page number, and an optional message (often the time and date). The user may also specify that the output go to a disk file (either console format or printer format). The ability to specify a disk file as output allows the user to convert LINEEDIT

files and BASIC program files into text files.

To invoke the program, first insert the utility disk into one of the disk drives. Then type the program name followed by the file name to be viewed (usual MDOS format). The file will then be displayed on the console. For example:

```
1:VIEW "1:NIMB"
```

will display the contents of the file NIMB. Both VIEW and NIMB were assumed to be on drive 1.

If it is desired to spool the output to the disk, an optional output file may be specified. The text sent to the output file will be exactly as it was sent to the console. For example:

```
1:VIEW "1:NIMB" "1:NIMB.TXT"
```

will produce an output file on drive one called NIMB.TXT which will contain the same text that is sent to the console (the file NIMB will be printed on the console at the same time as NIMB.TXT is being created).

Titled and numbered pages may also be sent to the printer. To send the output to the printer instead of the console specify the number (in decimal) of printable lines on a page. For example:

```
1:VIEW "1:BLACKBOX" 66
```

will specify printing of titled output on the printer with 66 lines to the page. As part of the titling, the program will automatically print the filename, page number, and a built in message (see below). As mentioned previously, an optional title may also be selected at this time. This is a good place to put the date and time the listing is being created, or any other pertinent information.

Additionally, when listing on the printer, VIEW keeps track of the number of lines printed so that even BASIC programs with 250 character lines are listed and paginated properly.

The disk file output option may be combined with the printing option to output the printing text, title and all, to the disk at the same time it is being printed. By assigning the printer off (ASSIGN 2 0), this option can be used to spool output to the disk for later printing. The output can then be printed later by attaching the printer ot the console (assign 2 3) and viewing the program to the console. If the date and time were entered in the optional message, then listings of several revisions of a program could be properly logged, without keeping a stack of paper.

Through the use of the disk file output option, BASIC programs and LINEEDIT files may be converted to plain text files. This is useful if an editor is available to edit them, or for use with other language systems. Modification of the program would allow LINEEDIT files to be created instead of straight text files. This would allow you to use the global search and replace features of LINEEDIT for BASIC programs, along with the use of TOKENIZE.

MTEST

MTEST allows the user to test memory for bad bits. The program is not a comprehensive test, but it will almost always find the chip which has kicked the bit bucket. The program is intended to test memory which has been operating properly and has then failed. These failures are more often of the hard kind (permanant and non-conditional) and are easily found.

To invoke the program, first insert the utility disk into one of the disk drives. Then type the program name, followed by the beginning and ending memory addresses of the memory area to be tested. For instance:

```
1:MTEST 2B80 FFFF
```

This will load the memory test program off of drive one and proceed to test the memory starting at address 2B80H and ending at FFFFH. Note that 2B80H is about the lowest memory address which can be tested because the memory test cannot test the memory that it occupies and must call MDOS to perform console I/O. To test below this address the program must be reassembled so as not to use these addresses and not call MDOS. Another possibility is to rearrange your memory configuration so that the memory not tested is swapped with the memory above 2B80H, which has been tested, and then to test this memory in its new upper addresses.

The test will run continuously without displaying any messages on the console until an error is found. The output format is with the numbers in binary (ones and zeros) to represent the bits of a byte in memory. The high order bit is on the left, low order on the right. Three numbers are printed which indicate: the starting increment of the test (I), the pattern read (R), the test type of short term retention (S) or long term retention (L) or complement (C), and the data written (W). An example error might appear as:

```
4C0F  I:00000001 RS:00000011 W:00000111
```

which indicates that at location 4C0FH on increment 1, the short term test read a different pattern from the written pattern.

ASCII MEMORY DUMP

The ASCII memory dump allows the user to dump memory in both hexadecimal and ASCII. When the program is executed, it will dump memory in the same manner as the MDOS DUMP command, except that the ASCII equivalent of the area dumped is printed to the right of the dump.

To invoke the program, first insert the utility disk into one of the disk drives. Then type the program name followed by the beginning and ending memory addresses of the memory area to be dumped. For instance:

```
1:ASCII 2A00 2B00
```

The program looks at the terminal width stored in MDOS to format the spacing on the dump. Thus the program will work equally well on a 64 or 80 column terminal without any changes.

The program $ASCII is exactly the same as ASCII except that it loads into one of the MDOS filebuffers and executes there. This allows the use of $ASCII to print a dump of an applications program which ASCII would normally overlay in part. Note that since this program does not load into the applications area, the MDOS APP command cannot be used to reenter it.

RECOVER

The RECOVER program allows the user to recover, on a sector by sector basis, sections of a diskette which have become unreadable by MDOS. When the program is executed, it will print the track numbers verified. If a sector is found which is difficult to read, the standard Micropolis retry procedure is followed.

If this fails, the track number (in hex) followed by a colon followed by the sector number (in hex) is printed on the terminal, and the sector is rewritten to the disk filled with zeros. If the bad sector is in the directory (track 0), the program wites the sector with an 0FFH pattern (this appears to MDOS as an empty directory). If the sector is number 3 of track zero, the DIR directory entry is written to give MDOS a directory to find. The program makes no attempt to restructure the directory, so this must be done by other means.

To invoke the program, first insert the utility disk into one of the disk drives. Then type the program name followed by the drive number which will contain the diskette to be recovered/verified.

For instance:

```
1:RECOVER 0
```

Note: RECOVER will also recover sectors on CP/M diskettes.

VERIFY

The VERIFY program allows the user to verify the data on a diskette on a sector by sector basis. When the program is executed, it will print the track numbers verified (in hex), sixteen sectors to a line, until all the tracks on a diskette have been verified. If a sector is found which is difficult to read, the standard Micropolis retry procedure is followed. If this fails, the track number (in hex) is printed on the terminal. When it has completed, it will print a message indicating the number of bad sectors found during this run.

To invoke the program, first insert the utility disk into one of the disk drives. Then type the program name followed by the drive number which will contain the diskette to be verified. For instance:

```
1:VERIFY 0
```

Note: VERIFY will also verify sectors on CP/M diskettes.

COMPRESS

The COMPRESS program allows the user to remove spaces and remarks from BASIC programs. The user is allowed to select the amount of compression desired. Spaces are always removed (no user option), but the user has the option of eliminating any combination of REM and ! comment statements. The program can even be compressed upon itself (into the same file), or rewrite an existing output file.

To invoke the program, first insert the utility disk into one of the disk drives. Then type the program name followed by drive number and name of the BASIC program file which is to be compressed (usual MDOS format), followed by the drive number and name of the file for the compressed output. For instance:

```
1:COMPRESS "1:NIMB" "1:NIMB.SQEZ" 0
```

will load the compress program off drive one and proceed to compress the file on drive one called NIMB. The 0 (zero) option will be described below. If the output file on drive one NIMB.SQEZ already exists, the program will prompt and ask if you wish it rewritten. An appropriate Y of N will continue or end the program.

There are four numeric options available in COMPRESS. They are:

0	remove all remarks
1	remove only REM statements
2	remove only ! statements
3	keep all remarks

TOKENIZE

The TOKENIZE program allows the user to convert an ASCII text file to a BASIC program file. The text in the file is searched for words which can be compressed into BASIC token. These words are looked up in a token table and sent to an output file as one character abbreviations for the word.

To invoke the program, first insert the utility disk into one of the disk drives. Then type the program name followed by drive number and name of the text file which is to be tokenized (usual MDOS format) and the drive number and name of the file where the tokens are to be placed. For instance:

```
1:TOKENIZE "1:NIMB.TXT" "1:NIMB"
```

will load the tokenize program off drive one and specify that the file on drive one called NIMB.TXT be tokenized. The output will be placed into the file on drive one called NIMB with a BASIC program file type. The file NIMB can then be loaded and executed by BASIC in the normal way.

Note that TOKENIZE does not check syntax.

....

GMS SOFTWARE

GMS has two utility disks for MDOS users. The File Utility Disk contains XFILES, FLIST, XTYPE, and LIST, all of which are general MDOS support routines. The Basic Utility Disk contains MAX-MIN, PLOADG, and BASPAC. All programs are discussed below.

Each disk is available from DAMAN for $38.50, postpaid. Add $7 outside North America. VISA and Master Card accepted. MOD I must add an additional $5 for convertion.

The GMS programs do not include source code, but do have a configuration routine which allows customization for your system.

GMS FILE UTILITY DISK

XFILES

XFILES is a MDOS command to list the file directory of a diskette with alphabetic type and flags plus decimal size. In addition to all files, a unique file name, all files of a type, or an ambiguous file name (for all files with a partially matched name) may be listed. Up to four ASCII parameters and/or up to four binary parameters may be specified.

An ASCII parameter is a name enclosed in double quotes and may specify: 1) a Unique File Name (UFN); 2) an Ambiguous File Name (AFN); or, 3) a file type. A binary parameter consists of a number from 0 to 3 representing a disk unit. Syntax examples:

```
XFILES "<drive>:<UFN>"
XFILES "<drive>:<AFN>"
XFILES "<drive>:$<type>"
XFILES <drive>
```

<drive> is the disk drive unit number. Drive 0 is assumed for ASCII parameters if it is omitted.

<UFN> a unique file name that does not have a '$' as the first character and does not contain any '?' or '*' characters.

<AFN> an ambiguous file name that may contain mask or match characters: '?' and '*' respectively. The '?' matches any single non-blank character, while the '*' matches zero or more characters to the end of a name, or to the next character specified.

<type> is a three character file type used by the file utilities. A '$' must precede the type, otherwise it is assumed to be a file name.

Parameter examples:

"$BAS"	all BASIC files on drive 0
"1:$ASM"	all ASM files on drive 1
1	all files on drive 1
0	all files on drive 0
"*"	all files on drive 0
"??"	all files on drive 0 with exactly two characters
"1:????*"	all files on drive 1 with four on more characters
"PROG??"	all files on drive 0 beginning with 'prog' followed by any two non-blank characters
"2:*.*"	all files on drive 2 containing a period anywhere
"A*B"	all files on drive 0 beginning with 'A' and ending with 'B' - just 'AB' would also be selected
"A*"	all files on drive 0 beginning with 'A' - also just 'A'
"*.A"	all files on drive 0 ending with '.A' - also just '.A'
"PROG"	the file 'PROG' on drive 0

Command example:

```
XFILES 0 "1:$BAS"
```

will display all files on drive 0 and all BASIC files on drive 1.

XFILES assumes all files on drive 0 if no parameters are specified.

The output is formatted with a symbolic name or decimal number for the file type and decimal number for the sector length. The permanent and write protect attributes are listed as P and W. The defined file types are listed under XTYPE.

Output example:

```
>XFILES "1:*.A"
XFILES 4 Copyright(c) 1982 GMS Software Ser#00-0000

Typ Siz 1:Filenm  Typ Siz 1:Filenm Typ Siz 1:Filenm
----------------  ---------------- ----------------
ASM 30P  XFILES.A ASM 85P  FLIST.A  ASM 66P  LIST.A
ASM 30P  XTYPE.A  ASM 79PW @EDIT.A

Selected 5 files, using 20 tracks.  19 tracks free.
```

"1:*.A" selected all files on drive 1 ending with '.A'. Five files were selected using 20 tracks among them. Drive one has 19 free tracks.

Configuration of XFILES is available by executing:

```
XFILES C
```

The configuration paramerters are: 1) # of files across the page; and, 2) # of spaces between columns. The configurator is self documenting.

FLIST

FLIST is a MDOS command to list the contents of all or selected ASM, BAS, or DOC files onto the printer. Headings and page numbers are added. Forms size is configurable. The command syntax is exactly the same as for XFILES. No default is taken: FLIST requires at least one parameter to produce output.

Example:

```
FLIST 0 "1:$BAS" "1:$ASM"
```

will list all ASM, BAS and DOC files on drive 0, all BAS files on drive 1, and all ASM files on drive 1. To prevent the listing of undesired files when selecting by type or AFN, check the files to be listed with XFILES using the same syntax.

The format of the heading is:

```
**** <file name> ****<optional user heading> Page
```

FLIST comes pre-configured for a page height of 66 lines and a page width of 132 columns. An instant confiurator is built into FLIST. Just sign on with

```
FLIST C
```

and FLIST will enter its configurator routine. This routine prompts the user for decimal configuration values. The configuration parameters for FLIST are (pre-configured values are in parenthesis): 1) page height in lines (66); 2) page width in columns (132); 3) initial top margin (1 for yes); and, 4) top margin in lines (4).

The initial top margin configuration parameter selects whether to output the first listing's top margin initially, or save it untill after the final listing; if done initially, the paper must be positioned with the print head at the form perforation. Printers with pin feed instead of tractor feed may have a tear bar a few lines above the print head. You can save a sheet of paper per listing by selecting the first top margin to be output after all listings are complete. Also, the top margin lines will have to be set equal to the number of lines from the print head to the tear bar.

The operation of the configurator is self documenting.

XTYPE

XTYPE is a MDOS command that sets the file type and/or permission attributes with alphabetic MUG command syntax. The format is:

```
XTYPE "<file>" "<attr>" "<type>"
```

The <file> specification and one or both of the other specifications are required.

<attr>
- P Make file permanent.
- W Make file write protected.
- N Remove both the P and W attributes.

The <attr> options are performed in the order of occurence. It is possible to remove one flag and set another by specifying the N option first.

<type> Symbolic file type from the following table, or a decimal number 0-63 (uses top 6 bits of the attributes byte).

TYP	DEC	HEX	DESCRIPTION
DAT	0	00-03	MDOS and BASIC data file.
ASM	1	04-07	Editor/Assembler source file.
OBJ	2	08-0B	Assembler object and BASIC save memory file.
OVL	3	0C-0F	System command (overlay) file, linkable from BASIC.
BAS	4	10-13	BASIC source file.
COM	5	14-17	System command file.
USR	6	1C-7F	User command file.
DOC	32	80-83	Document processor source file. *
ASC	33	84-87	CP/M ASCII format. *

* File types used by some GMS software.

The decimal type is the top 6 bits of the attributes. The hex type includes the permanent and read only bits.

Command examples:

XTYPE "PROG" "P"	make 'PROG' a permanent file
XTYPE "TEMP" "" "COM"	make 'TEMP' a command file
XTYPE "A" "PW" "COM"	make 'A' a permanent write protected command file
XTYPE "FILE" "N"	remove permanent and write protect attributes

LIST

LIST is a MDOS command to list a specified ASM, BAS, or DOC file onto the system console. Screen height and width are easily configured. The command syntax is:

```
LIST <file>
```

where <file> is an ASM, BAS, or DOC file.

If a pause is configured, LIST will fill the screen with the number of lines configured for the screen

height; otherwise continous scrolling of the file will occur. Press any character to get a new page after a pause, or enter ctl-c to cancel the listing.

An instant configurator is built into LIST which allows changing of screen height, output screen width, and the pause at configured screen height. The configurator is invoked by:

```
LIST C
```

It is self documenting.

CONFIGURING TYPES FOR XFILES, FLIST, AND XTYPE

The symbolic file types used by XFILES, FLIST, and XTYPE may be re-defined by modifying the file named TYPES.DEF, then invoking the types configurator MUG program: TYPCFG. Another program, TYPES, displays the currently configured types on your utility master.

Use ASSMEDIT to edit the file TYPES.DEF included on your File Utility Disk. There are 64 possible file types - each one must have exactly three characters. See the contents of TYPES.DEF for more information.

Assuming a copy of your File Utility Disk is in drive 0, perform the following sequence to redefine the file types:

```
>ASSMEDIT
LOAD "TYPES.DEF"
(CHANGE THE DESIRED TYPES)
RESAVE
DOS
>SCRATCH "TYPES.OBJ"
   (do this if TYPES.OBJ is present)
>ASSM "TYPES.DEF" "TYPES.OBJ" "EC"
>TYPCFG
TYPCFG 1 Copyright(c) 1982 GMS Software Ser#00-0000

0:FLIST:  configured.
0:XFILES: configured.
0:XTYPE:  configured.
0:TYPES:  configured.
```

Now, all of the file utilities on drive 0 have new types tables. TYPCFG expects the file TYPES.OBJ to be present on drive 0, but it will update the types tables for file utilities on other drives: just specify the drive # as the first binary parameter. EX: >TYPCFG 1 . (TYPCFG simply states 'not found.' for any of the four utilities that are not present).

....

GMS BASIC UTILITY DISK

MAX-MIN

MAX and MIN are programs which remove and restore the Micropolis BASIC 4.0 extended features EDIT, RENUM and MERGE while leaving the BASIC program residing in memory intact.

The BASIC 4.0 extended features EDIT, RENUM and MERGE, hereafter referred to as "features", are used frequently in program development. Sometimes a program and its run time variable space require the features to be removed so that there will be enough memory space left to run the program. Previously, one had to keep two versions of BASIC: one for running large programs and one for altering with the features. The full version, of course, is all that is necessary if the space used by the features is not needed. Micropolis provides a "FEATURES" program to remove the features; however, it deletes the current program. This is OK for creating the minimum sized BASIC needed for large programs. It provides an extra 1670 bytes over the maximum BASIC.

Without MAX and MIN, editing and re-running large programs which need the minimum BASIC requires:

(1) completely retyping the offending line(s)

or:

(1) save program
(2) link to full feature BASIC
(3) load program
(4) alter program
(5) save program
(6) link to minimum BASIC or FEATURES program
(7) load program

All of this is time consuming and creates unnecessary wear to the hardware and diskette surface.

With MAX and MIN, only the following is required:

(1) link to MAX to restore all features
(2) alter program
(3) link to MIN to remove the features

The top 2.8K of memory is used by MAX. MIN uses the top 1K. Five versions are provided as overlay modules only, for memory sizes of 32K to 60K. They are named MAXxx and MINxx, where xx is 32, 40, 48, 56 or 60.

The Max routine reloads the extended features object code and restores the names to the command table. Since the object code is part of a copyrighted software product of Micropolis, GMS cannot provide the complete MAX routine. The following shows how to merge the features object code with the applicable MAXxx routine supplied:

(1) boot your MDOS 4.0 system
(2) put a working copy of your Basic Utility Disk into drive 1
(3) enter: BASIC
(must be full feature version)
(4) SAVE "N:1:TEMP" 16R5700,16R5D85
(5) LINK "MDOS"
(6) LOAD "1:TEMP" yyyy
(7) enter: LOAD "1:MAXxx"
use the following table for xx and yyyy

SYS SIZE	xx	yyyy
32K	32	7700
40K	40	9700
48K	48	B700
56K	56	D700
60K	60	E700

(8) enter: SAVE "MAX" aaaa bbbb F cccc
use the following table for aaaa, bbbb and cccc

SYS SIZE	aaaa	bbbb	cccc
32K	7500	7D85	7500
40K	9500	9D85	9500
48K	B500	BD85	B500
56K	D500	DD85	D500
60K	E500	ED85	E500

(9) to get the appropriate copy of MIN
enter: FILECOPY "1:MINxx" "1:MIN"
where xx represents the system memory size
obtained from the above table

You should now have a version of MAX and MIN which will work on your system. From BASIC, type

LINK "MIN"

to remove the features, and

LINK "MAX"

to restore them.

For MAX you should get the following:

MAXxx 2 Copyright(c) 1982 GMS Software Ser #00-000

EDIT, RENUM and MERGE restored.
or:
EDIT, RENUM, and MERGE already present.

Where xx is the system memory size in decimal k.

MIN has a similiar sign-on line and produces one of the following messages:

EDIT, RENUM, and MERGE removed.
or:
EDIT, RENUM, and MERGE already removed.

An attempt to use MAX or MIN on other than BASIC 4.0 will result in the message:

System version error.

followed by a system soft halt.

If MAX or MIN are referenced as a commad while in MDOS, the message:

BASIC not loaded.

is displayed on the console.

PLOADG

PLOADG consists of system patches for BASIC 4.0 which allows specification of the BASIC program to be run each time BASIC is loaded from MDOS. ie. >BASIC "1:PROG". This is not for BASIC-only disks (disks that boot up to BASIC).

Installation is simple:
1) put diskette with MDOS and BASIC into drive 0
2) put the BASIC UTILITY DISK into drive 1
3) enter: PLOADG "1:1PLOADG.SAV"
you will get: 'Saving BASIC ...'

To try it out:

1) enter: LINK "MDOS"
BASIC "1:TEST"
you should get: 'TEST TEST TEST ...'

If your BASIC with PLOADG is ever re-saved by other than performing the steps given above, the PLOADG feature will not be present upon reloading BASIC. PLOADG disables the in-memory copy of itself each time BASIC is loaded, because it uses buffers which are used by BASIC. Therefore, if re-saving of BASIC is ever needed because of corrections of whatever, simply install your corrections and begin the PLOADG installation at step 2.

BASPAC

BASPAC is a MDOS user command to prepare BASIC programs for smaller memory size or vender program distribution by removing all "!" comments and insignificant blanks. "REM" comments, DATA statements and strings are left intact.

The command syntax is:

BASPAC "<file name>" <unit>

<file name> is any BASIC program file. <unit> is optional and specifies the disk unit to receive the compressed output. The output is written to the file "COMPRESSED".

....

CHEAP COMPUTER -- (Continued from column 2)

some sort of disk boot strap routine that SSM refers to cutely as their 'mystery bootstrap'.

The ultimate sadness is that to get the Micropolis up out of its chair it is necessary that you teach it to speak to the SSM monitor. Naturally everything is in the wrong register at the start. I wouldn't be surprized to discover that the monitor requires its input on 3 by 5 cards, inserted in some hidden slot.

When starting from scratch you actually have a tiny computer running just on the SSM cpu and video boards alone. This gets the keyboard input onto the video screen and, if you really don't know any better, it is quite an accomplishment to have come that far without some major catastrophe. But from that point on the going gets a bit sticky. It is necessary to get back to your Micropolis and patch the RES module to the proper SSM monitor addresses via... I say VIA untidy little routines to shuffle the registers about so that the Micropolis output is converted to the SSM-required output and visa-versa. It should take you about 20 minutes if you are clumsy with your hands.

Once you have patched into the system....I say, does any one really want to know all that? I'll be happy to tell you what worked for me, but I really think it is rather small beer; you only need the keyboard-output and video-input. The keyboard break routine is not vital and nothing else in the monitor is of any use whatsoever. It is only necessary in the first place because you have to talk to the thing SOMEHOW. Morse code dosen't work at all well and the only switches on my system are power on and reset. The monitor is only a necessary evil. Once you have access to your Micropolis software you can stuff the entire video routne into holes in the RES module. I think that is rather an elegant solution and I'll be happy to tell you how I did it just as soon as all the law suites are settled... or next letter, which ever comes first.

....

NOTES (Continued from column 1)

article - or a letter with questions. That's the only way you'll find out who are the rest of the members with the same interests. Articles are welcome in longhand, typed, or in MDOS LINEEDIT, EDIT/S, or CP/M ASCII format. You still get the $3 price for a library disk for each article (assuming you also sent a disk).

USER ASSISTANCE

I guess I've never said it in print, but questions or articles for sale are listed free in the classified section of the MUG newsletter. Longer questions are printed as letters to the editor.

....

MISCELLANEOUS MICROPOLIS INFO

by George W Shaw II
Shaw Laboratories, Limited
17453 Via Valencia, San Lorenzo, CA 94580

In scanning the news letters I received and other back issues I have been able to borrow, I have a few bits of trivia (or maybe not so) to share.

DISK MAINTENANCE

I completed the Micropolis class on micro-floopy servicing back in October of '78. One of the topics covered in the class was the spindle drive belt servicing. Mentioned in one of the MUG issues was that the belt should not be turned inside out, that is, that the smooth surface always faces the spindle pulley. This is important. Another consideration not mentioned is that the belt not be turned up-side down. In other words, the edge which faces the floppy drive should always face the floppy drive. The reason for this is that the belt may stretch or wear along one edge and thus cause the belt to slowly walk off the pulley if reinstalled the opposite way. Also, before reinstallation, the belt and pulleys should be cleaned thoroughly with alcohol and a Q-tip.

SAVING RES

Second, when making modifications to RES for the terminal or printer I/O routines, save them out on the disk before loading BASIC to test them out. BASIC overlays the portion of the code that RES

uses to load MDOS when the system boots off the disk. If you load BASIC first, and then later save RES on a disk you wish to boot off of, the system will not start up when that disk is booted because the code which loads MDOS is gone.

OSM

Lastly, I have brought up the Micropolis Micro-Disk with OSM on a couple of systems now and would be glad to share my knowledge with anyone who is having trouble. OSM is an extremely powerful multi-user/tasking operating system, much surpassing CP/M and probably MP/M. The BASIC on the disk is upward compatible with the MDOS BASIC. Additionally, the BASIC fully supports indexed files, eliminating the need for sorting in most applications. I also intend to do a version of A-FORTH for the Rigid-Disk.

....

LETTERS

DISKS

Buzz,
I've just learned something that perhaps all Micropolis users should know. It seems that our drives require the use of disks with reinforcing rings. This is common knowledge according to Shirley Mantz, sales representative for Dysan Co..

Shirley tells me that every Micropolis user that she has delt with use the reinforcing rings as a matter of necessity. Have you found this to be true? I've been using Micropolis drives for three years and I didn't know.

By the way, I'm using Dysan #105 / 1D diskettes and have been reasonably happy with them.

Bob Fortune
240 S. E. 8th St., Dundee Oregon 97115

Bob: I think you've answered your own question. Lots of Micropolis owners use single density disks without hub rings. I used the old Scotch single density for years, and most of my data disks are still the old Scotch, and running fine. However, some drives seem more finicky than others. When I started the Users Group Library Disks, some members couldn't read my disks. So I switched to Quad Density Verbatim disks which have ring reinforcements. I had no further complaints. Now I've switched back to Scotch double density, with reinforcements. I believe the single density also have reinforcements these days, too.

My point is, whatever works for you is OK. If you can read and write to a particular disk, I wouldn't advise spending additional money for anything more.

However, I'm told that ring reinforcement is more important than density. Ring reinforcement allows long term use - many more insertions and removals from the drive - a process that wears, or wringles the center hole of the disk. One is more likely to have a disk fail because it no longer centers properly than to have it fail because you wore the oxide off the surface.

In general, all manufacturers produce disks which should work for quad density drives. And, if general, a particular manufacturer makes all his disks the same way, there is no special process for making quad density disks, though different manufacturers do make different quality products.

Once he makes his product, the manufacturer only expects to sell a certain portion of that product as quad, or double, density, so he only tests, or certifies, a certain percentage for quad density, and another percentage for double density. But they are all out of the same mold. If the disks fail the multiple-density tests, they are relegated to single density - assuming the faults aren't severe enough to fail single density tests.

The biggest percentage of the production is only tested for single density, though. For this part of the production, neither you nor the manufacturer know whether they will pass a multiple density test. Chances are, they would. So, if you can get a box of single density disks for $40 and have one of them fail to initialize, you ended up paying $4.44 per disk for the good ones. That beats paying $80 for a guarenteed quad density ($8 each).

The method of certification varies for each vendor. Some vendors, such as Dysan, test every disk. Other vendors, say Scotch, only test a percentage of the disks - perhaps one out of ten, I really don't know the figure. The interesting thing is, Scotch makes the Dysan disks. The products are identical up to the point of putting disks in the box. Scotch then takes ten disks, tests one, and if it's OK, puts all ten in the box. Dysan tests all ten, and replaces any one that happens to fail. Again, check the prices and figure out whether the guarentee is worth the additional price. In reality, my personal failure rate for Scotch is much lower than one out of ten. It's more like one out of fifty doesn't initialize.

....

FORTH

Buzz,
I have another fix for the MUG FORTH. Line 630 in FORTHSOR3 should be changed from

```
    DB  'O'+80H
         TO
    DB  '0'+80H
```

That's change 'letter oh' to 'number zero'. Thanks go to Tom Ceska for finding this typo.

Richard Newman

....

MICROPOLIS BASIC MANUAL

Buzz,
Could you recommend, or do you have, a manual on Micropolis Basic programming?

John Casey
3005 Julia St., Tampa FL 33609

John: There isn't any Micropolis Basic programming manual. For general Basic programming, I recommend "Basic and the Personal Computer", by Dwyer & Critchfield, published by Addison-Wesley. It is good, covering the generalities of statements first, then going into arrays, sorting, graphics, data bases, and simulations. It really takes you from "knowing nothing" to a fairly advanced level. The exception is disk file manipulation. No coverage here. I've written a couple articles (March & April of 82). Perhaps we need some more.

....

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) 883-2621

HUNTSVILLE ALA. MAY 28'82 U.S. POSTAGE 0.20 PB METER 1343328

FIRST CLASS MAIL
===== ===== ====