MICROPOLIS USERS GROUP

MUG Newsletter # 19 - February 1982

****

THE MUG IS A S/W HOUSE

This month has been a very busy one. Setting up the capability for handling S/W is no easy task. No single distributor handles all the products a buyer wants. In addition, there is a lot of good software that is distributed by the individual developers.

Three considerations led me to the decision to handle software. One is that writing this newsletter is not a money-making situation. I wanted to find a means of paying for my time. Increasing the price of the newsletter didn't seem reasonable at this point, though it's possible in July. SOLOS, the Processor Technology (SOL) newsletter, charges $24/$32 for about the same idea.

Secondly, a lot of people want information and reviews on the available software packages. Since one has to buy it and use it to review it, I figured that by becoming a dealer I could cut my costs for obtaining information for this sort of story.

Finally, the concept of a users group providing software at a discount, and providing the knowledge necessary for a user to ascertain which of a number of competitive packages is correct for his needs, was attractive to me.

I decided to use the DAMAN name for servicing the retailing of software, rather than the MUG. As I've said before, I've run a mailing-list/invoice business, out of the house, under the name of DAMAN for several years. The selling of software will also be run out of the home.

Between Lynn, Brad and myself, we hope to develop a complete understanding of most of the packages we handle. Either Lynn or Brad should always be around during the day, and I'm around at night.

How does this all affect the newsletter? There will be more articles on specific packages, such as the one on Systemation's UTL-1 and TR/II in this issue. Now, however, in addition to informing you, my desire is also to have you purchase the package from me if you decide that it fills your needs.

What software do we offer? Almost anything is available. The price list shown in the back pages only lists those packages I've researched. Many other packages are available through the distributors with which I'm dealing - such as the full lines of Digital Research and Microsoft.

If you are considering the purchase of any software, give us a call. We can probably get it for you at a below-list cost. The percentage discount varies. But as you can see from the price list, some are large. The popular SPELLBINDER is only $319 (list at $495), and dBASEII is $589 (list at $695).

Most of the CP/M software is also available in formats other than Micropolis, if you happen to have other computers around, or for your friends who have other computers.

....

DIMENSIONING IN MpBASIC

by Buzz Rudow

Dimensioning is the process of setting aside memory space for variables used in your program. MpBASIC performs some default dimensioning by itself.

Generally, reference to any single variable, such as, A$, A, A1, automatically allocates space for that variable without using the DIM statement. This automatic allocation is caused by the default value of the SIZES statement.

String variables, like A$, are allocated a 40 character maximum width. Therefore, without any further specific dimensioning, if you try and put information of greater than 40 character size in A$, the information will be truncated at 40 characters. You won't get any error - the process of truncation is a legitimate and useful operation. You can, of course, put in fewer than the maximum width. If you want a string variable to contain more or less than the default 40 characters, you must specifically dimension it. For example -

```
    DIM A$(80)
      or
    DIM A$(5)
```

DIM A$(80) or DIM A$(5) will allocate A$ an 80 character or 5 character width. If you have several string variables which must contain different widths, each must be dimensioned.

```
     DIM A$(80),B$(5),C$(32)
             or
     DIM A$(80)
     DIM B$(5)
     DIM C$(32)
```

There is no physical requirement on where to put a dimension statement, but it must logically be executed before any reference is made to it in your program. Dimensioning does not have to be done in one place. Different variables can be dimensioned where and when you use them.

Once dimensioned, you can't change (redimension) its width in the program.

If you want to change the string default value of 40 characters in width, use the SIZES statement. Default SIZES is SIZES(5,3,40) where the 3rd number specifies the width of all strings which are not specifically dimensioned. You can make the maximum size of all such strings smaller, say 15 characters, by executing the statement -

```
     SIZES(5,3,15)
```

(See newsletter #2 for a discussion of the first two numbers in the SIZES statement)

STRING ARRAYS

An array is a data structure which enables the programmer to loop through a set of items by calculated, or pointed, reference, rather than specific reference. Arrays are referenced by array variable name and location. A$(3) is the 4th location (remember, zero is legitemate - 0,1,2,3) of the A$ array. Understand that A$ is distinctly different from the array variable A$. You can have A$ and A$(0) in the same program.

Consider verifying a legitimate state abbreviation. Suppose A$ contained some string which is suppose to be a legitimate abbreviation. One might write it -

```
800 IF A$="AL" THEN GOTO 1000
810 IF A$="AK" THEN GOTO 1000

      (48 similar statements)

890 IF A$="WY" THEN GOTO 1000
900 PRINT "ERROR - STATE NOT CORRECT"
910 GOTO SOME ERROR ROUTINE
1000 !CONTINUE - VERIFIED OK
```

Or, one might set-up an array of 51 legitimate state abbreviations. The dimension statement would be -

```
     DIM B$(50,2)
```

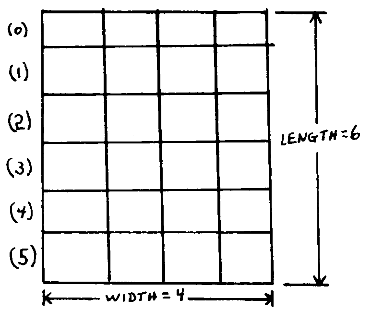
where '50' is the length and '2' is the width. The length is physically 51 because '0' is a legitimate location. If this array were initialized with the 51 legitimate state abbreviations, then the previous example could be written -

```
800 FOR I=0 TO 50
810 IF A$=B$(I) THEN GOTO 1000
820 NEXT I
900 PRINT "ERROR - STATE NOT CORRECT"
910 GOTO ERROR ROUTINE
1000 !CONTINUE - VERIFIED OK
```

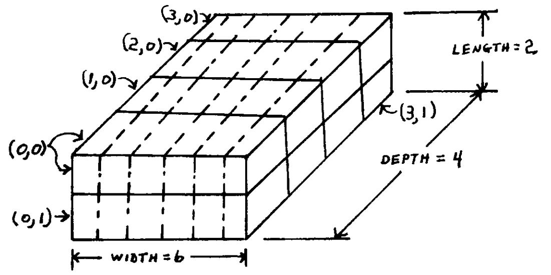
Even though you used two numbers in the DIM statement, this is called a "single-dimension" array. The dimension is the length, or '50' in this case. When referencing the data in the array, you specify its position. B$(0) is the value of the first string at the first location, B$(10) is the value of the eleventh string, and B$(I) is the value of the "i"th string.

The length of an array can be set from 0 to 60 - some thousand, though you'll get a memory overflow with large numbers. The width can be set from 1 to 250. You must set a width. An array does not default to 40 character width if the second number is left off. In fact, if you leave the second number off, you haven't defined an array, you defined the width of the lone B$ as 50 characters.

A single-dimensioned string array, such as DIM C$(5,4), can be visualized as:



You can also have multiple dimensioned arrays. The doubly dimensioned string array DIM C$(3,1,6) can be visualized as:



It has a depth of 4, a length of 2 and with each string having a maximum character length of 6. When you reference this structure, you use two numbers, C$(i,j), where 'i' is the depth position and 'j' is the length position.

Typing in, running, and analyzing the following program will hopefully finalize the idea of what dimensioning of strings is all about.

```
10 DIM S$(3,1,20)
20 FOR A=0 TO 3
30 FOR B=0 TO 1
40 READ S$(A,B)
50 NEXT B
60 NEXT A
70 DATA "ALABAMA","MONTGOMERY","ALASKA","JUNEAU"
80 DATA "ARIZONA","PHOENIX","ARKANSAS","LITTLE ROCK"
90 PRINT TAB(4);"STATE CAPITOLS"
100 PRINT
110 PRINT "STATE","CAPITAL"
120 PRINT
130 FOR A=0 TO 3
```

```
140 FOR B=0 TO 1
150 PRINT S$(A,B)
160 NEXT B
170 PRINT
180 NEXT A
190 END
```

....

MICROPOLIS PDS DESCRIPTION

by Buzz Rudow

As in most cases, there are exceptions to the rules, but in general, the Micropolis PDS (Program Development System) can be broken down into 3 areas. These are RES (RESident routines), an executive, and application programs. Executives and application programs change, but RES stays in the computer.

RES is the accumulation of resident system routines which are used by the executive, and which can be used by the application program. The RES routines perform the input-output (I/O) functions, such as keyboard to memory, disk to memory, memory to screen, and memory to disk. If you have other devices (collectively called peripherals) connected to your system, then their I/O will also be serviced by RES. Other peripherals might include printers, modems, and tape drives.

RES is the one part of the PDS that has different code for different systems. It has to be configured for the specific hardware on which it is operating. RES is physically located in the low part of memory, normally from 02B0H to 1598H.

The primary Micropolis executives are named MDOS (Micropolis Disk Operating System) and BASIC (Beginner's Allpurpose Symbolic Instruction Code). Other programs function as executives, but we'll get back to that later.

Most systems have RES configured so that a "cold boot" brings in RES, which then brings in MDOS off the disk. One can build a BASIC only system. That is one in which the "cold boot" brings in RES, which then brings in BASIC. The point is, BASIC is an independant executive, not an application program of MDOS. Most users feel, because BASIC is normally loaded from MDOS, that BASIC is an MDOS application program. Not so.

Another unique feature of RES is that when you first turn on your computer and bring up PDS, you, in one way or another, transfer control to the ROM on the Micropolis disk controller. The program in ROM will read RES from the disk and write it into the low part of your memory. Although most disk reads are made by name, the transfer of RES is an exception. The ROM program reads whatever is in track one of your disk. It could be named WHOCARES and it still would read into memory. RES has to be in track one. Actually, whatever the boot load is, has to be in track one. Normally, that is RES.

The point to be made here is that when making a system disk, always place RES on the disk as the first file. If you aren't ready to place RES, you can leave room for it by putting on a dummy file. This is done via MDOS by executing -

```
     CREATE "SPACER"
```

The SPACER file can later be scratched and replaced with RES.

The MDOS executive contains expanded disk I/O routines and a bunch of subroutines for manipulating numbers and characters. It does not have the capability of constructing application programs. The construction must be done by way of an editor, such as LINEEDIT. The program then has to be assembled before MDOS can execute it. MDOS can load and transfer control (execute) to an application program which was previously constructed. The application program will use routines in MDOS and MDOS will use routines in RES.

MDOS is physically located at the end of RES, normally from 1598H to 2B00H.

The BASIC executive is similar in some respects to MDOS. It also contains expanded disk I/O routines and a bunch of subroutines for manipulating numbers and characters. BASIC, however, does have the capability of constructing its applications programs. BASIC can execute its programs without assembling or compiling them first.

....

FIXING PROGRAMS THAT HANG

by Buzz Rudow

One of the most exasperating situations to be in is to have a program "hang". It won't run to completion, but it doesn't abort. So you sit there and wait - and wait, not wanting to press CONTROL C because, maybe, just maybe, it's doing what it's supposed to do.

Finally your patience wears out and you press CONTROL C. The program stops and there is a line displayed on the screen. What do you do now?

Actually, you can do just about anything except edit the program. You'll still be able to continue running the program where it has just stopped. Editing is the insertion, addition, deletion, or modification of any line in the program. Don't type EDIT (or RUN), and don't type anything that has a numeric digit as the first character. If the first character is a number, BASIC thinks you are adding a line, even if you didn't mean to.

Micropolis BASIC is an interactive language. This means, in part, that it allows you to type in and execute most of the BASIC commands without giving line numbers and running a program. You can LIST or LISTP the program to see what you are working with.

I would normally analyse the listing in the area of the line number which was displayed when the CONTROL C was pressed. I figure out what is supposed to be happening. I can determine that certain files are intended to be OPEN, and certain variables should contain numbers or text which will enable me to see where the program is in its execution. You can PRINT any of these variables. I would probably PRINT my loop variable (the (I) in the loop FOR I=1 to 1000), and the parameters associated with the disk files.

You can set any of the variables, too. If you wanted to increase the loop counter to jump it across bad data at, for instance, record 56, you type

```
     I=57
```

The point I'm trying to make is that you can do anything from the keyboard in "real-time", by typing the BASIC statement without a line number. You can do GOSUBs, GOTOs, PUTs and GETs, etc. You can examine, alter, and correct your program before you command it to continue its automatic execution.

To continue execution, type

```
     CONT
```

THE WORST KIND OF 'HUNG'

Now suppose we go back to the assumption that this program you're running is hung and you have decided to press CONTROL C. Last time it stopped. This time - NOTHING HAPPENS!!! You are stuck in some input/output loop that isn't looking for operator interruption by CONTROL C.

Most of you know that the only thing to do is RESET your computer. It is likely that most of you would also REBOOT MDOS, BASIC, and reload your application program. However, if you have not turned off power, you don't have to do a COLD BOOT.

Do a WARM BOOT. From your Monitor, transfer control to 04E7H. On my SOL, that's EX 4E7. On the VG it's G 04E7.

You'll find that BASIC is still there - the program is still there - in fact, all the variables are there with the values they contained when you RESET. Files are still open, and you can now manually close them and save all that good data. You have all the capability to examine that you had when Control-C worked.

If some of you CP/Mers know of a similar entry point for BASIC-80, let me know about it.

....

CONVERSION BETWEEN MDOS & CP/M

by Buzz Rudow

For those of you who aren't content to operate in one system, there's a relatively easy way to convert BASIC, SOURCE, and DATA files between the MDOS and CP/M systems. Conversion is necessary because the systems structure their directories and their files differently. The tie to Micropolis is maintained by the way the data is placed on, and removed from, the disk. Both MDOS and CP/M have almost identical code for the "primitive" disk access routines, which, more or less, means the way a sector is located, read, or written. Each system's interpretation of the sector's data is different, however.

Suppose you have a CP/M BASIC-80 program you want to run under MpBASIC. A four-step process is required.
(1) Save the BASIC-80 program in ASCII form.
(2) change the CP/M ASCII to MpASCII.
(3) Change MpASCII to MpBASIC.
(4) Change the syntax to suit MpBASIC.

I tried this using a BASIC-80 game, BUZZWORD. It has a catchy name. To make it ASCII, I saved it as SAVE "BUZZWORD",A, - the "A" suffix doing the ASCII task. Here's the program as a BASIC-80 CP/M file.

```
5 !REM TEST OF SYSTEMATION CONVERSION, CP/M TO MDOS,
  12/23/81
10 PRINT CHR$(4):WIDTH 80
20 PRINT TAB(26);"BUZZWORD GENERATOR":PRINT
30 PRINT TAB(15);"CREATIVE COMPUTING MORRISTOWN, NJ"
40 PRINT:PRINT:PRINT
50 PRINT "THIS PROGRAM PRINTS HIGHLY ACCEPTABLE
   PHRASES IN"
60 PRINT "'EDUCATOR-SPEAK' THAT YOU CAN WORK INTO
   REPORTS"
70 PRINT "AND SPEECHES.  WHENEVER A QUESTION MARK
   IS PRINTED,"
80 PRINT "TYPE A 'Y' FOR ANOTHER PHRASE OR 'N' TO
   QUIT."
90 PRINT:PRINT:PRINT "HERE'S THE FIRST PHRASE:"
100 DIM A$(40)
110 FOR I=1 TO 39 : READ A$(I) : NEXT I
120 PRINT A$(INT(13*RND(1)+1));" ";
130 PRINT A$(INT(13*RND(1)+14));" ";
140 PRINT A$(INT(13*RND(1)+27)) : PRINT
150 INPUT Y$ : IF Y$="Y" THEN 120 ELSE GOTO 260
160 DATA "ABILITY","BASAL","BEHAVIORAL","CHILD-
    CENTERED"
170 DATA "DIFFERENTIATED","DISCOVERY","FLEXIBLE",
    "HETEROGENEOUS"
180 DATA "HOMOGENEOUS","MANIPULATIVE","MODULAR",
    "TAVISTOCK"
190 DATA "INDIVIDUALIZED","LEARNING","EVALUATIVE",
    "OBJECTIVE"
200 DATA "COGNITIVE","ENRICHMENT","SCHEDULING",
    "HUMANISTIC"
210 DATA "INTEGRATED","NON-GRADED","TRAINING",
    "VERTICAL AGE"
220 DATA "MOTIVATIONAL","CREATIVE","GROUPING",
    "MODIFICATION"
230 DATA "ACCOUNTABILITY","PROCESS","CORE CUR
    RICULUM","ALGORITHM"
240 DATA "PERFORMANCE","REINFORCEMENT","OPEN
    CLASSROOM","RESOURCE"
250 DATA "STRUCTURE","FACILITY","ENVIRONMENT"
260 DATA PRINT "COME BACK WHEN YOU NEED HELP WITH
```

```
ANOTHER REPORT!":RUN "MENU"
```

CP/M TO MDOS FORMAT CONVERSION

Then I rebooted, getting out of CP/M and into MDOS. Next I ran a Systemation utility, named CP/M-MDOS. The following is an exact reproduction of the dialog.

```
>CP/M-MDOS
CP/M-MDOS Utility - Rev. #1.0 - Serial #17157
Copyright (C) 1979 by Systemation, inc.

Enter drive and name of CP/M source file (cr)
?B:BUZZWORD.BAS

Enter drive and name of MDOS destination file (cr)
?0:BUZZWORD

Pack sectors 2/1 (Y or N) ?Y

Insert formatted diskettes in specified units -
Type Y when ready - ? Y

As the destination file was generated in 16K
blocks, it could occupy more disk space than the
source file. However, if any sectors were
appended, they are merely filled with NUL (Binary
0), and may be deleted with MDOS or BASIC.

Copy completed -- BYE ! !
```

MDOS ASCII TO MpBASIC CONVERSION

At this point I have a MDOS file of TYPE 0. The next step is to use another Systemation routine, TR/II, to change the ASCII into MpBASIC format. As shown by Burks Smith last month, that requires the changing of key words to tokens. TR/II expects any ASCII file to be a TYPE 84, which is accomodated by executing -

```
TYPE "BUZZWORD" 84
```

Then TR/II was invoked. The following is the dialog.

```
>TR/II "BUZZWORD" "BUZZWORDB" 10

Translator II - BASIC/ASCII - Rev. #1.00 - Serial
#12106 Copyright (c) 1980 by Systemation, inc.

Translation completed successfully !

Happy Programming !
```

CORRECT THE SYNTAX

Simple. OK - ready to run. The program is now executable, though it has syntax errors for MpBASIC. One finds syntax errors by looking at the printout or by running it and changing lines when the computer halts at an illegal syntax.

This particular program required the following edits.

```
Line 10 : Delete WIDTH 80, change CHR$ to CHAR$
Line 100 : Change (40) to (40,20)
Line 120, 130, 140 : Change RND(1) to RND(0)
Line 150 : Delete ELSE GOTO 260
Insert new line 155 : GOTO 260
```

To convert an MDOS program to CP/M, just perform the same steps in reverse. Change MpBASIC to MpASCII, MpASCII to CP/M ASCII, and correct the syntax.

```
TR/II "BUZZWORD" "BUZZWORDB" 84
MDOS-CP/M
Enter drive and name of MDOS source file (cr)
?0:BUZZWORDB
Enter drive and name of CP/M destination file (cr)
?B:BUZZWORD.BAS
```

I also converted some LINEEDIT files to CP/M with the same set of programs. This would allow you to use the data with CP/M wordprocessors such as SPELLBINDER and WORDSTAR.

If you're interested in such things, the required programs are available through the MUG.

TR/II list $55,	for the MUG $49
UTL-1 list $95,	for the MUG $85

UTL-1 contains the CP/M-MDOS and MDOS-CP/M routines, as well as 7 other utilities. See newsletter 5, pg. 2, for further discussion of UTL-1.

These prices are POSTPAID to N. America. Add $7.50/package elsewhere. VISA and MASTERCARD accepted. Phone orders, (205) 881-1697.

....

COMMERCIAL (???) SOFTWARE

An interesting new program has come to my attention. It will appeal to only a limited set of people, but might prove very useful to those who frequent the race tracks. The MUG has no knowledge of its capability, and is making no guarentees.

PONY - PICK

If you are a thoroughbred racing enthusiast, a professioal handicapper or just someone who likes to win more than they lose, then PONY-PICK is the computer handicapping program you need.

Unlike others that have been written in the past, PONY-PICK is not sensitive to any one track or any one type of race. PONY-PICK can be used for almost any race on virtually every track in America. And, PONY-PICK the only system that tells you just how accurate it thinks it is.

The secret of its accuracy is the use of ARTIFICIAL INTELLIGENCE in performing its forecast. Simply, it learns about your favorite tracks from the data you enter for each racing day. This allows it to "Fine-Tune" itself to these tracks and their specific racing conditions. Moreover, it's able to pick up the short term variables, (new jockeys, etc.) that can make every horse run slightly out of form and consider these in its forecast.

Because it adjusts itself automatically and is constantly striving to improve it's accuracy, it advises you just how accurate it has become and permits you to change your wagering strategy accordingly. And, if for some reason its accuracy is down, it lets you know that too! Try to find that in another program!

PONY- PICK requires a minimun system of 48K, dual-drives, CRT with 80 character screen and a printer capable of 80 characters/line. PONY-PICK was written by BONJOEL Enterprises and is supplied in object code. It runs under the Systemation, Inc. RUN/S or RUN/Z Run Time software. PONY-PICK is available from the MUG for $252 (list $300). RUN/S or Z sell for $58 (list $65) in the Rev. 2 version.

....

AUTO CONFIGURATION

by Buzz Rudow

There really are two different problems involved in auto configuration, meaning the ability of software to run on any system. One is to get a group of software items that are on a disk to operate on any one member's single computer without the member having to do complicated edits to all the programs. The second is to have all that same group run on any computer.

This situation is a result of getting inputs from many members. Our library disks have clear screen commands that sometimes read PRINT CHAR$(4), OR CHAR$(11) or CHAR$(27)+"+" - or any of literally dozens of other variations. The reason for the confusion is that video display manufacturers have not standardized their control characters.

To cure the first problem, either you or I have to find the clear screen command in each program and change them so that they are correct for our own computer. This "configuration" can be done one time, saved, and then never bothered with again.

It can, as long as you don't want to run the software with a different terminal. There are some of us who have more than one kind of computer systems. If I configure for my SOL, the software won't run on my Vector Graphic or CCS. I would need three different versions of each program - multiple disks - etc. That's a pain, and expensive in terms of time and money.

The solution to this second problem is either to have the software recognize which computer it is running in, or have a menu in which the name of the system is requested.

My initial work was done with the menu concept, coupled with a spin-off of Dave Land's configuration routines (March MUG Newsletter). In response to the menu, the program POKED a number into location 165 (A5H). Each program controlled by the menu then PEEKed location 165. Based on its value, one of a set of pre-defined statements was executed, which placed the proper character codes in a variable array chosen as O$(n). If limited to the clear screen command, the variable was only O$(0).

This solution assumes that location 165 is free for all members, which may not be the case. The subroutines also can get very long when all the various terminals are considered. Finally, the most troublesome problem was that I would have to edit each and every program in the library whenever a new terminal came along.

As far as the software determining what system it's running in is concerned, one could do this for his personal group of computers. There is data that can identify the computer. Then, taking me as an example, I know what peripheals I have tied to my Vector Graphics, SOL, and CCS. There isn`t any data in the operating system or montitor that tells me what peripheals are there. However, I just know that I have a Televideo on the CCS. Someone else could have a Hazeltine 1500. So, as I said, this idea works OK for an individual, but not for the Group`s library disks.

The solution I`ve settled on for the group is this. Instead of having a subroutine, I PEEK the actual characters to be used in the clear screen command. This takes a few more bytes of memory, but the benefits are worth it. No more subroutines on each program that have to be updated, and no longer do those subroutines take up valualbe program space.

The next question is where do I PEEK, and how does the proper data get in there. This time I took an idea from Jerry Factor (MUG newsletter #4) about saving variables in the Micropolis BASIC prompt space. Here are 42 bytes of space that are free, and, I hope, are in the same location for all of us. The data is put in through a statement in a master MENU.

This puts one piece of the work on you. You have to edit one line on each disk. I`ll even give you instructions on what to put in.

There are three other enhansements that will be incorporated. I put a version of the INKEY routine in the same 42 bytes. It only takes 21 bytes, so we still have 7 left after INKEY and the other configuration data. This INKEY waits for any keystroke. Youll have to test for validity of the keystrokes in your program. However, for any single character response, Y(yes), N(no), C(continue), P(print), etc., you don`t need to press RETURN. Sometimes it's confusing to have some responses require using RETURN and some not. I hope to be able to mod the programs so that any single stroke doesn`t require a RETURN, but multiple strokes do. The system will even deduce whether it's running on Ver. 3 or 4 of BASIC.

Secondly, I added two bytes for specifying the

default drive for programs and data.

The final enhansement is useful only to the MOD I people. On a MOD II, each master MENU will call two secondary menus, which in turn call multiple programs. The MOD Is will have the MASTER and one of the secondaries on a disk. This should ease the problems I`ve had in converting the MOD II disks to MOD Is. Besides, each menu will control similar programs, so this is structured programing.

Here's what a Master Menu will look like:

```
10 ! 11/14/81 MENU
11 J%=16R2F06
12 IF PEEK(16R04C9)=64 THEN J%=16R2F7A
13 IF PEEK(16R04C9)=64 THEN DEF FAA=16R2F80
14 IF PEEK(16R04C9)=0 THEN DEF FAA=16R2F0C
20 DIM O$(1,8)
40 IF CHAR$(PEEK(J%+37))<>"/" OR CHAR$(PEEK(J%+40))<
>"/" THEN GOSUB 59015:GOTO 60: !Configure system
50 GOSUB 57115: ! Read Clear Screen
60 GOSUB 40015: ! Process MENU
70 END
40000 !
40005 ! 11/14/81 Main Menu
40010 !
40015 PRINT O$(0)
40020 PRINT TAB(21);"MICROPOLIS USERS GROUP"
40025 PRINT TAB(21);"LIBRARY DISK 08, REV 00"
40030 PRINT TAB(28);"MAIN MENU"
40035 PRINT
40040 PRINT "0 - EXIT LIBRARY DISK 08"
40045 PRINT "1 - APPLICATION & UTILITY PROGRAMS"
40050 PRINT "2 - GAMES"
40055 PRINT
40060 PRINT "Enter Number of Function Desired. ";
40065 A$=FAA(1)
40070 IF A$<"0" OR A$> "2" THEN GOTO 40065
40075 PRINT A$
40080 IF A$="0" PRINT O$(0):PRINT "MICROPOLIS USERS
GROUP LIBRARY DISK 08 EXITED.":GOTO 40105
40085 OPEN 8 "DIR" ERROR 40115
40090 CLOSE 8
40095 IF A$="1" PLOADG "MENU.A"
40100 PLOADG "MENU.G"
40105 RETURN
40110 !
40115 PRINT
40120 PRINT "******";ERR$;"******"
40125 PRINT "Correct Problem, Press RETURN to Contin
ue.";
40130 A$=FAA(1)
40135 GOTO 40085
57000 !
57005 ! 11/14/81 Read Date
57010 !
57015 O$(1)=""
57020 FOR N%=1 TO 8
57025 O$(1)=O$(1)+CHAR$(PEEK(J%+34+N%))
57030 NEXT N%
57035 RETURN
57100 !
57105 ! 11/14/81 Read Clear Screen
57110 !
57115 O$(0)=""
57120 FOR N%=0 TO 2
57125 O$(0)=O$(0)+CHAR$(PEEK(J%+N%))
57130 NEXT N%
57135 RETURN
59000 !
59005 ! 11/14/81 Configure software for system
59010 !
59015 FOR N%=0 TO 26
59018 READ I%
59021 POKE(J%+N%)=I%
59024 NEXT N%
59026 ! Data for Clear Screen
59027 DATA 32,32,11,48,49,0
59030 DATA 62,3,50,160,1,62,1,50,161,1,50,162,1,205,
123,7,120,50,163,1,201: ! Data for INKEY
59033 GOSUB 57115: !Read Clear Screen
59036 PRINT O$(0)
59039 FOR N%=1 TO 6
59042 PRINT
59045 NEXT N%
59048 PRINT TAB(10);"Your System has been Configured
."
59051 FOR N%=1 TO 200
59054 NEXT N%
```

```
59057 PRINT O$(0)
59060 FOR N%=1 TO 6
59063 PRINT
59066 NEXT N%
59069 PRINT TAB(20);"Good Morning!!"
59072 FOR N%=1 TO 200
59075 NEXT N%
59078 GOSUB 59115: !Set Date
59081 RETURN
59100 !
59105 ! 11/14/81 Configure date for system
59110 !
59115 PRINT O$(0)
59118 FOR N%=1 TO 6
59121 PRINT
59124 NEXT N%
59127 PRINT "Please enter Date (MMDDYY) for use in T
oday's Programs."
59130 INPUT O$(1)
59133 IF LEN(O$(1))<> 6 PRINT "**ERROR**":PRINT "PLE
ASE ENTER 6 DIGIT NUMBER":GOTO 59127
59136 FOR N%=1 TO 6
59139 IF MID$(O$(1),N%,1)<"0" OR MID$(O$(1),N%,1)>"9
" PRINT "**ERROR**":PRINT "ALL SIX CHARACTERS MUST B
E NUMERIC":GOTO 59127
59142 NEXT N%
59145 O$(1)=LEFT$(O$(1),2)+"/"+MID$(O$(1),3,2)+"/"+R
IGHT$(O$(1),2)
59148 FOR N%=1 TO 8
59151 POKE(J%+34+N%)=ASC(MID$(O$(1),N%,1))
59154 NEXT N%
59157 PRINT O$(0)
59160 GOSUB 57015
59163 PRINT TAB(10);"THE SYSTEM DATE IS: ";O$(1)
59166 PRINT
59169 PRINT TAB(10);"Is this correct? (Y) or (N) ";
59172 R$=FAA(1)
59175 IF R$<>"Y" AND R$<>"N" THEN GOTO 59172
59178 PRINT R$
59181 IF R$="N" THEN GOTO 59115
59184 RETURN
```

You'll have to edit line 59027 which, in this case, is set up for a SOL. This is from Library Disk 8, which incorporates the concept.

Lines 11-14 deduce whether the system is Ver. 3 or 4 of BASIC. J% is set to the starting location of the 'Micropolis BASIC' prompt which we are going to overwrite. Line 40 checks for the slashes. If they are already in memory, then you have already configured, and the program falls through to line 50.

If the slashes are not there, subroutine 59015 reads the data values at lines 59027 and 59030 and puts the values in the prompt area. Line 59027 is the Clear Screen commands (3 bytes max), two bytes for selecting default program and data drives, and one byte left for a spare.

If your clear screen command is less that three bytes then the leading bytes should be 32, or an ASCII blank. The drive bytes are 48 for zero, 49 for one, which are ASCII '0' and '1'.

Line 59030's data is the decimal representation of an INKEY routine similar to the one I wrote in newsletter #9. The subroutine at 57100 reads back the clear screen data you just wrote to the lower memory. Subroutine 57100 is repeated in each program on the disk. We have configured once, and the data will remain in low memory as long as BASIC stays in memory. The subroutine at 59100 asks for the current date. It sets the slashes in low memory which cause this whole routine to be skipped the next time you execute line 40. Subroutine 57000 reads the date back in for verification. If needed in print routines, the subroutine is repeated in any such program on the disk.

Line 40015 shows my standard way of using O$(0) for clearing the screen. All programs use O$(0). Lines 40065-40075 illustrate the use of the INKEY routine. The system will only recognize a '0', '1', or '2'. Line 40070, or its equivelent, is rewritten to mask out illegal responses. Lines 59169-59178 show the same idea for a Yes/No response.

OK, there you have it. Let me know if I've made some errors in my judgement. I hope any of you who obtain disk-8 will let me know if the ease of operation is worth my task of setting up all the programs. Well, actually, it wasn't my task - my son Brad did all the work.

....

LETTERS

Buzz,
I have been trying to get up a FORTH system for the Micropolis from the FIG-FORTH model. Frankly, I have had problems in debugging the program. Most of it is ok, as I have checked it repeatiably with the published code. I have a fair amount of time into this project already as the source is 80 pages long. I would be interested in corresponding with others with an interest in putting FORTH on Micropolis.

In lieu of a self-made system, I bought and am currently running a Z-80 FORTH ($50.00) from Laboratory Microsystems (unfortunately it runs under CP/M and not MDOS but at least he can sell it to you on Micropolis Mod II format diskettes). To give you some idea of the speed of FORTH, when I ran Jim Gilbreath`s benchmark program on prime numbers (Byte, Sept. 81) my 2 MHz Sorcerer executed the program in 15 seconds. A 4 MHz Z-80 will run the benchmark in 7.5 seconds, which is only 0.7 seconds slower than the Z-80 assembly language routine listed in the Byte article and between 2 to 100 times faster than any other language tested (on a 4MHz Z-80). I should point out that the Laboratory Microsystems FORTH is optimized for the Z-80 and is not an 8080 program like the FORTH program`s tested in the article. Anyway, for a real time programming envirnoment FORTH is definitely way ahead of everyone else.

I am enclosing some "fixes" for the Sorcerer computer running MDOS version 4.0. They originate from Exidy but I got them through a dealer (Ed Mentzner of Mentzner Electronics). These patches eliminate problems with early carriage returns, erase rubbed out characters instead of printing an underline on the screen, and telling MDOS Basic to stop grabbing memory before it overruns the monitor stack area. All of the patches work as I have implimented them and have had no problems to date.

I am also enclosing the details for modifying a Sorcerer I or II computer so that the Micropolis controller can reside at FC00 where normally the user graphics are placed. This mod involes one trace cut, adding one 74LS21 chip, and soldering about a dozen connections. This allows the user the option of having a full 48k of memory, use of a ROM PAC of 8K additional memory in the S-100 expansion box, and use of the disk all at the same time, without the disk controller taking any memory space. The modification allows the user to switch back to the `normal` mode if he/she wants to run a game which uses the graphics area. I also have details on modifying the Micropolis disk controller for those 48K Sorcerer owners who would like to put the controller at BC00 so they can use the disk and a ROM PAC (only a trace cut and a couple of wires to solder). I will be happy to send out copies to interested Sorcerer owners, if they care to write me.

Dr. Richard S. Neuman
Faculty of Medicine, Memorial University,
St. John`s Newfoundland A1B 3V6 CANADA

....

CLASSIFIED

FOR SALE: Processor Tech "SOL" Computer with SOLOS monitor. S-100 system with 56K RAM. Micropolis MOD II disk drives, controller and software. CP/M and T-MAKER II. Complete System $2,200 or parts. Call evenings "CST" (608)-788-6677.

Pete Eversole

....

DAMAN
604 Springwood Cir.
Huntsville AL 35803
(205) 881-1697

==

Software Price List - 02/01/82
MDOS MDOS MDOS MDOS MDOS

==

AFB MICRO CONTROLS

	LIST	SALE
Complete Attorney's Business Pkg..........	$495	XXX
Invoice-Writer...........................	275	XXX
Medical/Dental...........................	495	XXX
Mfg. Order-Entry/Inventory Cntl...........	495	xxx
Payroll/Job Cost.........................	275	xxx

BONJOEL

DATABASII	Data Base Generator............	$ 50	$42
MODFILE-			
MODIMATH	Aux Pkg for DATABASII...........	50	42
INVEN-1	Small Business Inventory Mgmt...	50	42
WAMSORT	Assembly Language Sort..........	40	34
REACT	Calendar Reminder...............	50	42
PONY-PICK	Thoroughbred Handicapping.......	300	252

CHAMELEON SOFTWARE

GAME 1	Balrog Sampler..................	$ 30	$ 27
GAME 2	Stone of Sisyphus...............	30	27
GAME 3	Morton`s Fork...................	30	27

DAMAN

CATALOG I	MDOS Disk Cataloging...........	$ 30	$ 22
MAILSYS I	General Mailing System..........	50	37
MEMSYS I	General Group Membership System.	50	37

DATASMITH

G/L	General Ledger.................	$250	$189
PAYROLL	Payroll........................	350	263
MULMERGE	Merge BASIC into multiple pgms..	30	27
SMASH	Reduces BASIC pgm size..........	30	27
SYSLIST	List multiple BASIC pgms........	30	27
TEXTCONV	BASIC to LINEEDIT and back......	75	60
VARLIST	Lists BASIC variables & arrays..	30	27
	Set of 5 above utilities........	150	116
DATAMGR	....	450	398

GMS SOFTWARE

MDOC	Document Processor in BASIC.....	$ 75	$ XX
MAX-MIN	Remove/Restore FEATURES.........	30	XX
GAMEDISK	Blackjack, Biorhythm, Banner & Lucas..................	35	XX
FLIST	LIST Multiple ASM or BAS........	25	XX
BASPAK	Deletes ! comments & blanks.....	30	XX
XFILES	Directory in alpha order* size..........................	5	X
XTYPE	Sets attributes*................	5	X
PLOADG	Direct boot to BASIC program....	30	XX

* with any other GMS order

LENZ SOFTWARE

DBANK	Disk Banking (PER or BUS).......	$ 75	$ 62

MONK SOFTWARE

BEST	Custom Building Estimation......	$295	$231

ORGANIC SOFTWARE

TEXTWRITR	Text Formatter..................	$125	$110

SYSTEMATION

AUTO/EXEC	System Generator...............	$ 40	$ 36
BASIC/S	Extended BASIC Compiler.........	345	308
BCOMPARE	Basic Comparison................	35	31
BEM	Basic Expansion Module..........	65	58
CRUNCH	Basic Compactor.................	35	31
DSM-1	8080-8085 Disk Disassembler.....	65	58
EDIT/S	Text Editor.....................	45	40
RUN/S	BASIC/S Run-Time Package only...	65	58
SORT/A	Hybrid Sort (for Mp BASIC)......	75	67
TR/II	Translator II - BASIC/ASCII.....	55	49
UTL-1	Disk Utility Package............	95	85
XREF	Cross Reference Generator.......	85	76

Prices and availability subject to change without notice.

==

Software Price List - 02/01/82
CP/M CP/M CP/M CP/M CP/M
on Micropolis Formatted Disks

==

ASHTON-TATE

	LIST	SALE
dBASE II................................	$695	$589

BONJOEL

PONY-PICK Thoroughbred Handicapping.......	$300	$252

CHARLES MERRITT

SPELL MENU..............................	$ 95	$ 72

COMPILER SYSTEMS

CBASIC2.................................	$150	$126

COMPUTER HEADWEAR

WHATSIT?................................	$150	$131

BUSINESS PLANNING SYSTEMS

PLAN 80.................................	$295	$253

DATASMITH

G/L	General Ledger..................	$250	$189
PAYROLL	Payroll.........................	$350	$263

INNOVATIVE SOFTWARE

SPELLGUARD..............................	$295	$253

INVESTMENT SYSTEMS ANAL

PROPERTY MANAGEMENT SYSTEM II.............	$725	$690
PROPERTY ANALYSIS SYSTEM..................	250	242
BUSINESS SUPPORT SOFTWARE.................	65	65

LEXISOFT

SPELLBINDER.............................	$495	$319
SPELLCHECK..............................	$295	253

ORGANIC SOFTWARE

MILESTONE	$295	$253
DATEBOOK II	295	253
PERSONAL DATEBOOK	150	131
TEXTWRITER II or III	125	110

SORCIM

ACT I 8080/Z80	$175	$161
PASCAL/M	395	358
SUPERCALC	295	253

SYSTEMATION

BASIC/Z	Extended BASIC Compiler	$345	$308
RUN/Z	BASIC/Z Run-Time Package only	65	58
SORT/B	Hybrid Sort (for BASIC-80)	75	67
UNDELETE	File Recovery	45	40
UNPROTECT	Basic Source Recovery	70	62

TECHNICAL SOFTWARE

PASCAL SORT	$195	$169

TOPAZ SOFTWARE

SBASIC	$295	$253

Prices and availability subject to change without notice. Most CP/M software available in formats other than Micropolis.

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

HUNTSVILLE ALA. JAN 27'82 U.S. POSTAGE 0.20 P.B.METER 1543328

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