NOTES - MICRODOT MANUAL UPDATE - August, 1989

New MULTI Module Version

The MULTI module is now 1.3. Version 1.3 fixes a bug which could cause MicroDot to lose its place if you were reading a directory, then opened another directory and read from it, then returned to reading the first directory.

New GARB Module Version

GARB version 1.2 collects garbage quite well. It collects it so well that it trashes most of your strings -- OUCH! The GARB module has been updated to version 1.3 to fix this bug. For you assembly-language hackmeisters out there, the problem was that I left off the ",x" on an instruction that was supposed to be indexed. A one-byte change was all that was needed to fix the problem.

New Program Writer Adapter

MicroDot's Program Writer Adapter is now version 1.2. The change was made to support Beagle Bros' new version of Program Writer, which comes on a disk named /PROGRAM.WRITER/ rather than /EDITOR/. The Adapter now asks you what slot and drive the Program Writer disk is in. It also asks you what slot and drive to save the patched Program Writer files to -- there's no longer room on the MicroDot master disk to hold the patched Program Writer.

The new PW.ADAPTER also fixes an obscure bug which sometimes caused the computer to crash after editing a program with Program Writer. The bug was caused by mis-set zero-page variable pointers.

Other Stuff

All other MicroDot files other than , MULTI, GARB, & PW.ADAPTER are unchanged from version 1.2. This INCLUDES MICRODOT.SYSTEM, which is still version 1.2.

NOTES - MICRODOT MANUAL UPDATE - March, 1989 (1.2 notes also apply to 1.3, except as detailed above)

New MicroDot Version

MicroDot is now version 1.2. Previous versions of MicroDot had bugs in the &.BO command that caused a ?SYNTAX ERROR. The problem was that &.BO was looking for an extra comma before the number. For example, you might type &.BO,F$,#1000 but MicroDot would issue an error because it wanted another comma after the # sign: &.BO,F$,#,1000. This has been fixed. Accordingly, MicroDot's page 3 version number now stands at 2.

&.BO Enhancement

Since I was fixing &.BO anyway, I decided to enhance it to better support libraries of commonly used subroutines. On page 38 of the manual, I said: "WARNING: THE &.BO COMMAND DOES NOT CHANGE THE END OF PROGRAM POINTER! YOU WOULD HAVE TO DO THIS MANUALLY IF YOU USE THIS TECHNIQUE." Well, you don't have to anymore. When you use the # option of &.BO to attach an overlay to a program in memory, MicroDot will adjust Applesoft's end-of-program pointer to include the loaded overlay. When you type &.S to save your program, the new module will be saved with it. This is the case whether you use &.BO from immediate mode or in a program.

If you type &.BO in immediate mode, the LOMEM pointer will be moved to the end of the newly loaded module; if you use &.BO in a program, LOMEM will stay where you left it. Remember, LOMEM is where your variables are, not necessarily where your program ends. (MicroDot does it in immediate mode to keep your variables from accidentally overwriting part of your program; in a program, you still need to manually set LOMEM so that there's enough room for your modules below the variables. MicroDot assumes you know what you're doing, so it doesn't try to outguess you.)

CAT80 Enhancement

CAT80 has been enhanced to better handle the creation and modification time fields in the directory listing. Times are now shown in AM/PM format, and times with hours of zero are now correctly interpreted to be midnight. Times with minutes of zero are now correctly shown in the listing.

CLOCK.FIX Enhancement

If you use a IIgs, you don't need the CLOCK.FIX module. The new CLOCK.FIX automatically detects the IIgs; if a IIgs is found, CLOCK.FIX is not installed.

If you don't have a clock in your system, CLOCK.FIX will automatically install a "phantom clock." This "clock" doesn't keep real time; it simply advances one minute each time you issue an &.S or &.BS (or &.IS) command. This will at least allow you to tell which files on your disk are the most recent. Be sure to use the SET.DATE program to set the proper date.

The phantom clock is actually installed into ProDOS and remains installed even if you quit MicroDot. It will not cause any compatability problems with other programs, but other programs might not cause the clock to advance properly.

Bugs Fixed in Disk Formatter (HFORMAT)

Hyper.FORMAT, the disk formatter upon which MicroDot's disk format routines are based, requires the use of memory addresses $6710 to $800F as a track buffer. (This is fundamental to the design of Hyper.FORMAT and could not easily be changed.) To allow you a little more freedom in how you arrange your programs' memory, the MicroDot formatters save this area of memory elsewhere before formatting a disk, then restore before exiting. For example, FORMAT.1 saves the contents of the track buffer to $2800 before formatting, then restores that memory after formatting is complete. This allows you to put something else important at $6700 and up (such as your BASIC program!). (And that's the reason, by the way, why you shouldn't hit reset while formatting a disk!)

However, due to a programming oversight, MicroDot's formatters (under MicroDot 1.0 and 1.1) only saved $6800 through $7FFF. Thus, there are 256 bytes which are trashed by the formatter: $6710-$67FF, and $8000-$800F. The new programs, HFORMAT.1 and HFORMAT.2, on the version 1.2 disk, fix this. However, due to memory conflicts, I had to use the keyboard buffer to save the extra 256 bytes during formatting. This means that every time you format a disk using HFORMAT.1 or HFORMAT.2, the keyboard buffer will be trashed, which means that you can't format a disk from immediate mode with HFORMAT.1 or HFORMAT.2.

The memory conflicts with FORMAT.3 (the version which resides at $6000) were unsolvable since part of the program lies inside the area which is trashed. Therefore, there is no HFORMAT.3 on this disk.

New Disk Formatter (QFORMAT)

A completely different disk formatter which neatly solves the memory problems associated with Hyper.FORMAT's huge track buffer is included on the disk as QFORMAT. QFORMAT is short for Quick's Format, which has nothing to do with its speed -- it was written by Shawn Quick of QuickSoft. In fact, QFORMAT is about the same speed as Apple's disk formatter, which means it is somewhat slower than HFORMAT. It also takes up a little more space on the disk. However, since QFORMAT doesn't need a large memory buffer, it's actually more compact than HFORMAT. QFORMAT is completely self-contained and uses no memory outside the program itself, except for $2E0-$2FF, the very tail-end of the keyboard buffer.

In addition, QFORMAT verifies each track as it is written, so that disk drives which are too slow or too fast can be detected. HFORMAT doesn't do any such verification, which is one of the reasons it's so fast. QFORMAT can also be Reset with no ill effects.

QFORMAT is actually a hybrid program: the 5.25" disk formatting routine is by Shawn Quick; the rest of the program is the same as HFORMAT. Please note that QFORMAT only affects the formatting of 5.25" disks; formatting of 3.5" disks remains the same.

You can substitute QFORMAT into existing MicroDot BASIC programs with no changes to the program itself. Since QFORMAT is more reliable and uses less memory, I encourage its use instead of HFORMAT.

SoftWorks Included!

The MicroDot disk now includes Mark Munz' SoftWorks 4.1, an ampersand-driven program which allows you to add AppleWorks-style screens, filecards, and menus to your BASIC programs. It runs under MicroDot OR BASIC.SYSTEM. The docs are on disk in an AppleWorks Word Processor file. SoftWorks is freeware, but is copyrighted.

Compatibility with 1.1

MicroDot 1.2 is completely compatible with MicroDot 1.1. All the 1.1 modules will work with MicroDot 1.2, and vice versa. However, all of the modules have been changed a little bit (the changes generally don't affect their function, only their size, which is smaller, partly due to an improved relocation routine), so I would reccommend using 1.2 version modules.

Manual Errors

There are a couple of errors in the MicroDot manual. The table beginning at the bottom of page 55 (and ending on page 56) has incorrect PEEKs. They should be:

640	Storage Type/Name Length
641-655	Filename as returned by &.G
656	Type as returned by &.G
657-658	Pointer to key block
659-660	Number of blocks used by file
661-662-663	Length of file in bytes
664-665	Creation date
666-667	Creation time
668	Version of ProDOS which created file
669	Minimum version of ProDOS which can access file
670	Access control bits ($C3/195= unlocked, $01 = locked)
671-672	Auxtype as returned by &.G
673-674	Modification date
675-676	Modification time

On the back of the command summary sheet, the formats for all the date and time fields are shown as YYYYYYYMMMMDDDDD. Obviously this is incorrect for time fields. The correct format for a time field is HHHHHHHHMMMMMMMM as shown in the manual on pages 55 and 56.

Note that HHHHHHHHMMMMMMMM does not mean that the hour is in the first byte and the minute is in the second byte. Since the 6502-series microprocessors used in the Apple II store the low order byte of a number before the high order byte, the minute is in the first byte while the hour is in the second. The order of date fields is likewise reversed. This isn't an error in the manual but a result of the way ProDOS and the Apple do things.

NOTES - MICRODOT MANUAL UPDATE - December, 1988 (1.1 notes also apply to 1.2, except as detailed above)

New MicroDot Version

MicroDot is now version 1.1. Some minor changes were made to the way memory is allocated for optional modules. This was done to make it easier to write more MicroDot modules in the future. The result of this is that 1.0 modules will not work with MicroDot 1.1 and vice versa. Also, the value of VERSION (in MicroDot's page-3 area) is now 1, not 0 as it was previously.

Startup Menu

As stated in the manual, the startup program on the MicroDot disk automatically installs CLOCK.FIX (if you have a clock installed) and CAT40 (or CAT80 if you have 80-column capability). However, if you press the space bar while booting the MicroDot disk, a menu will appear to allow you to install your choice of modules with a single keypress. You can also run the disk formatter, execute the Program Writer patch program, and install the line editor from this menu.

Fast Garbage Collector

The standard Applesoft garbage collector is notoriously sluggish. Since BASIC.SYSTEM has a built-in fast garbage collector, we decided to include one with MicroDot in the form of an optional module called GARB. GARB can be installed like any other module with the &.BX command (see the Optional Modules secion in the manual). GARB is based on Bill Basham's public-domain garbage collection routine, which was included with Diversi-DOS and was published in Open-Apple in March, 1985.

Once GARB has been installed, its operation is fully automatic. To force garbage collection, include a statement like X = USR (0) in your program. Like X = FRE (0), X = USR (0) collects garbage and tells you the amound of unused memory, except that USR uses the fast garbage collection routine. (Also, GARB's USR always returns a positive number of bytes, unlike FRE, which returns a negative number if you have more than 32K free.)

MicroDot's garbage collector is not quite as fast as the one in BASIC.SYSTEM, although it is a great improvement over naked Applesoft. BASIC.SYSTEM's FRE requires a 1K memory buffer besides the actual garbage collection routines; MicroDot's routine requires fewer than 700 bytes.

Program Writer Patcher

A new version of the Program Writer patcher is also included with this version of MicroDot. It's faster than the old one, taking at most a minute to patch Program Writer. It also prompts you to insert the correct disk at the proper time, in case you happen to be stuck with one disk drive. In addition, you can now use the Program Writer configuration program on patched versions of Program Writer with the new patch program. There's no need to configure a unpatched version of Program Writer and re-patch it.

You should NOT try to use MicroDot 1.0 versions of Program Writer with MicroDot 1.1 because of the differences in memory allocation