PicoDOS8 User's Guide

Features

- Included with Persistor CF8 purchase
- Supports all Standard C Library File System Functions
- Supports TxBASIC with CFSave, CFRead, and CFExec Extensions
- Supports both 256K and 1MB Tattletales
- 100% DOS/Windows compatible media format
- DOS-like Command Shell (DIR, REN, ERA, COPY, FORMAT)
- Run or Chain Applications and Batch Files (with arguments)
- XMODEM/YMODEM (batch) offload/download (to 230,400 baud)

About the PicoDOS8 User's Guide

PicoDOS8 is our DOS-like operating system for the CF8/TT8 combination that provides both a command line user interface for common card and file operations as well as the underlying DOS FAT file system. It's this that lets your C and BASIC programs easily create and manipulate files that can later be read directly by your PC using inexpensive flash memory card readers. Here you will find descriptions of how to use the DOS-like commands and details of how PicoDOS8's presence impacts the amount of TT8 onboard flash and ram memory usage available to your applications.

Other Documentation

Getting Started Guides

This is the key must-read document if you're to have successful experiments with the Persistor. There's a separate Getting Started Guide for each of the various Persistors and it's one of the only two printed documents that come with your Persistor. If you don't have this on hand, the latest version is always available from our www.persistor.com web site and you'll find PDF and html copies on the installation diskette. If you haven't read and worked through the installation procedures, do that first before attempting to use any of the features described in this guide.

Persistor Data Sheets

This is the other printed document that comes with your Persistor and describes the electrical, mechanical, and environmental specifications you may need to design your experiment.

PicoDOS8 TxBASIC Guide

The PicoDOS8 TxBASIC Guide shows you how to use the Persistor and PicoDOS8 to save acquired data, stored in TxBASIC datafiles, to Windows compatible files on the flash memory card.

PicoDOS8 C Programming Guide

The PicoDOS8 C Programming Guide shows you how to adapt your C programs to take advantage of standard ANSI C file system capabilities that become possible with the addition of a Persistor and memory cards. This guide assumes that you are familiar with C programming and the ANSI C file functions, while describing the subtle differences between coding for desktop applications with unlimited processing power, memory space, and disk speed and coding for embedded applications where processing power is limited, battery power is precious, and file I/O takes considerably longer.

PicoDOS8 In Action

PicoDOS8 was developed to solve the two main problems that probably prompted you to buy the Persistor product in the first place: Getting data onto non-volatile storage media (data logging), and getting it back off (recovery and analysis). For the most part, if you've worked with files on a DOS machine, then you already know how to work the flash memory card in a Persistor. In addition, and again for the most part, the files on the flash card are transportable in both directions between PCs with card readers and the Persistor.

We've provided a couple of examples to show how simple it is to update your logger/controller to take advantage of high capacity, non-volatile storage.

Have you run the logger.c example that ships with the TT8 software? Take a look at our modified logcf8.c project that adds 9 lines of new code to convert from a RAM based logger storing 100,000 samples, to a non-volatile logger storing up to 150 million samples.

Remember the adventure game Zork? It feeds from a 133KB database file and is based on C source consisting of over 35 source files that are now in the public domain. We've included this as an example to stress test PicoDOS8-and because we like to brag that it took only three new lines of code to compile and run this under PicoDOS8!

You probably didn't buy a Persistor to build games, and though you may use it to update legacy applications, the real benefits to this product will come as you rethink your logger and controller applications in the context of the new capabilities added by the Persistor and PicoDOS8.

PicoDOS8 Command Shell

The PicoDOS8 command shell does several things. First, and foremost, it provides command handlers for common DOS directory manipulation functions. In addition, it provides facilities to move files back and forth between a host computer through a serial link, and acts as a launch pad to run applications and batch files direct from the flash memory card. You also get functional equivalents for all of the standard TOM8 monitor commands. Finally, it provides the low level file services required by your running applications.

The command shell and PicoDOS8 kernel live at the top of flash and take away about 98KB of flash and 28KB of high RAM, leaving about 160KB flash and 220KB RAM for your application's use. If that seems like too much, remember, you'll get this back and more with the new ability to chain to programs on the flash memory card, and maybe some of those giant tables you have coded into static data structures are candidates to become disk files that load to RAM at runtime.

You give a little, you get a lot. For example, with PicoDOS8, you can break up that huge monolithic program into separate modules: Setup, by interacting with a user and storing run-time parameters to a disk file. Acquisition, by using the disk file parameters to control the sampling. Recovery, by perhaps attempting some on the fly data compression.

For existing programs that do not reach into the top of flash and that do not make calls to PicoDOS8 (they couldn't 'cause it didn't exist!), the presence of PicoDOS8 in flash will have absolutely no effect. As described ahead, you must initialize PicoDOS8 with a call to InitCF8() to activate the kernel, and even then, it does nothing between calls for service.

For new or updated programs that wish to take advantage of PicoDOS8 services, nothing really changes in the makefiles or build processes you've been working with. The mechanics of PicoDOS8 are encapsulated in tiny inline functions defined in <PicoDOS8.h> and dispatched through inline TRAP12 instructions, all of which you can blissfully ignore as you'll be programming file operations at a much higher level using the standard C library functions.

The following information is provided for the insatiably curious, and for those folks who might be poking into undocumented reserved RAM areas or actually using 68000 TRAP 12 instructions. If that's not you, you can safely skip ahead to the next section.

Flash Memory Map

```
000000 - 0017FF    TOM8 Monitor
001800 - 001FFF    Custom TPU Code
002000 - 027FFF    Your flash application
028000 - 03FFFF    PicoDOS8 kernel
```

RAM Memory Map

```
2C0000 - 2C1FFF    Application stack
2C2000 - 2F7FFF    Application code
           [ram'd] followed by heap
2F8000 - 2FEFFF    PicoDOS8
2FF000 - 2FFBFF    -- unknown --
2FFC00 - 2FFFFF    Vector Base
            (PicoDOS8 @ TRAP12:2FFCB0)
```

TOM8 Commands

The PicoDOS8 command shell does not replace the TOM8 monitor; it continues to live at the base of flash, and you can always get back to the TOM8 from the command shell by typing TOM8, from your applications with the ResetToMon() function, and from hardware by grounding IRQ3 at reset or power up.

The command shell does however replicate all of the TOM8 functions so that you can conveniently autoboot to PicoDOS8 and still continue to use the same tools you're used to without having to swap back and forth between PicoDOS8 and the TOM8 monitor and without having to make any changes to your makefiles or build utilities. The TOM8 commands include the following.

```
?    ?
G    g [address]
GO   go [address]
LO   lo [ofs][;Bx[+]][;G]
MD   md [address]
MM   mm [address]
```

DOS Commands

To minimize your learning curve, and wherever possible, the PicoDOS8 shell commands mimic the syntax and behavior of their DOS counterparts. If you're not sure about a command, make a guess and try it (well, maybe not erase!). These standard DOS commands and their usage are summarized ahead.

```
COPY    copy source destination
DATE    date
DEL     del filename
DIR     dir [wildcards]
ERASE   erase filename
FORMAT  format [/Q][/E][/F]
TIME    time
TYPE    type filename
REN     oldname newname
VER     ver
```

Unlike DOS, and with the exception of the DIR command, all of the filename specifications must be complete unambiguous filenames without a drive letter (no "A:"). To erase all of the files, you have to either manually work through all the files, or re-FORMAT the card. The DATE and TIME commands both invoke the same handler and are a bit different from their DOS namesakes, but they are compatible with CrossCut and MotoCross, and get the time and date into the TT8.

FORMAT - Format a flash card

Format with no optional switches prepares the partition, boot, and directory sectors for DOS compatibility. In doing so, it effectively gives you a clean slate from which to begin recording, but it does not actually erase the data sectors themselves.

The /E switch forces PicoDOS8 to clear every sector to all FFs. This puts additional wear and tear on the card, but it does guarantee that on a catastrophic

failure, any data found on the card was put there after the last format. The The /Q switch tells PicoDOS8 to be quiet during the format (no messages). The /F switch puts the card into high current mode for the duration of the format to speed up the operation. This is most useful when used in conjunction with the /E switch.

File Commands

Though you can directly transfer files between the flash memory cards and a DOS computer with PC Card slots using the PC Card adapter, it's sometime more convenient to just transfer the files serially. The PicoDOS8 command shell supports XMODEM (128, 1K, and CRC) and YMODEM (including batch) file transfers in both directions, and at rates up to 230,400 baud.

```
BAUD     baud [newrate [/Q]]
CCC      ccc
CAPTURE  filename [delim] [/N]
DUMP     filename [startofs[,endofs]]
SAVE     filename [startaddr, endaddr]
XS       xs [/Q] [/X] [/C] filename
XR       xr [/Q] [/X] [/C] [filename]
YS       ys [/Q] [/G] filename [,file2...]
YR       yr [/Q] [/G]
```

BAUD - Change the baud rate

When you request a new baud rate, PicoDOS8 prompts you to change your host baud rate then hit a key to continue. The TT8 can handle any standard rate from 75 to 230,400, though most PCs can only go up to 115,200. PicoDOS8 always reverts back to 9600 baud at reset, though you can make an autoexec.bat file entry to force any other rate at startup.

Typing baud by itself displays the current baud rate which is rarely exactly what you specified, instead being the actual closest baud rate the hardware can produce.
If you specify the optional quiet flag (-Q or /Q), there will be no messages or prompts. This is primarily for use in batch files where interaction may not be desirable.

CCC - Cooperative Card Change

The CCC command puts the TT8 into LPSTOP mode until the next keystroke. In this mode, it may be possible to remove or insert a flash memory card without bombing the TT8. No guarantees, but it usually works for us, and you may also find this convenient.

CAPTURE

The CAPTURE command copies everything coming in from the serial port into RAM until you send a break character (the default), or until it sees the specified delimiter (use 3 for CTRL-C). It then copies the whole thing to the filename you specified. It buffers into RAM from 2C2000 to 2F800 which gives you about 250KB to work with.

Using TYPE with CAPTURE is a convenient way to edit small batch files with most modern terminal programs (like CrossCut or MotoCross). TYPE the current file from PicoDOS8, select and copy the text from your terminal program, paste into an editor and make the changes, select and copy again, the paste the update back after issuing the CAPTURE command. It's actually easier than it reads.

The option /N switch causes capture to append linefeed characters to incoming carriage returns. If your captures don't display correctly when you TYPE them back, try using this switch.

SAVE

The SAVE command copies a block of memory from the TT8 to a file on the flash memory card. This could be used to capture the results of a RAM logging session, but is more likely used to move programs loaded into RAM onto the flash memory card for later execution. If no start and end address parameters are provided, SAVE uses the information from the last LO command (or from MotoCross high speed loads) to get the address range. Do not run a program in RAM prior to saving it. It may work, but you should expect some bizarre results. You can run the program immediately after saving if you like.

DUMP - Dump file in Hex and ASCII

The DUMP command displays the contents of the specified file as lines of hexadecimal numbers followed by the mating ASCII characters (where valid). You can also specify an optional starting and ending offset into the file where display should begin and end.

XS and XR

The XS (XMODEM Send) and XR (XMODEM Receive) both initiate an XMODEM session at the TT8 end, and it's expected you will take the appropriate action at the host computer end. Both of these implement a smart XMODEM protocol hunt and should synchronize with hosts that support original 128 byte block XMODEM, CRC, and 1k XMODEM. You can force them to first try original XMODEM with the -X or /X option, or XMODEM 128CRC with the -C or /C option.

If you don't specify a filename, the XR command names the incoming file RECEIVE.NNN where NNN is the next available extension number. What the host does with the XS command filename will vary from machine to machine. CrossCut prompts you for a destination filename.

YS and YR

The YS and YR commands are similar to the XS and XR commands, but they support batch file transfers with automatic file naming using the more modern YMODEM protocol. This is the preferred protocol if supported by your host (sorry, Crosscut doesn't talk YMODEM) for speed and convenience. YS and YR default to normal YMODEM batch, but can be forced to YMODEMG (no error checking/recovery) with the -G or /G option. -Q inhibits the routines from emitting messages.

Behavior Commands

The remaining PicoDOS8 commands allow you to save and run TT8 programs stored on flash memory card or otherwise control TT8 execution.

```
BOOT    boot [P...] [T...]
RESET   reset
TOM8    tom8
```

BOOT

The BOOT command with no arguments displays the current bootup operation which will be either to the TOM8 monitor, to PicoDOS8, or ??? for some unknown application currently in flash. Specifying P as the first command character inserts a tiny jump program into flash at location 0x2000 which tricks the TOM8 into automatically running PicoDOS8 at power up or reset. Use this when you get tired of typing G 28000. You can clear this by typing BOOT TOM8 at the PicoDOS8 prompt. The boot to PicoDOS8 automatically gets cleared when you load your own application into flash.

RESET

The RESET command causes a complete system reset, which will either enter the TOM8, run an application in RAM, or put you back into PicoDOS8, depending on the current environment. This is the same a calling Reset() from inside your application.

TOM8

The TOM8 command forces execution back to the TOM8 monitor, whether or not there's an application burned in flash, and independent of the BOOT setting. This is the same a calling ResetToMon() from inside your application.

Command Files

Like DOS, PicoDOS8 allows you to load and run applications stored on the flash memory card which have the filename extension .RUN. The .RUN files we use, are the same .RUN files created by MotoCross or the Aztec compiler when you use the standard Onset makefile templates (the .RUN files are the .RHX files before being expanded to HEX format).

You can get .RUN files to the flash memory card by using the standard CrossCut or MotoCross methods to load a RAM application, but type SAVE filename.RUN <return> instead of G (MotoCross users will have to backspace first). If you did this with a built version of the logger.c example, and saved it to LOGGER.RUN, you can just type LOGGER<return> and you'll be running the logger example. You can also transfer existing .RUN files to the flash memory card using XR and Crosscut (or YR and something else), then SAVE the load with the extension .RUN.

Older applications designed for the TT8 had no way of knowing they'd be running under an operating system, so they couldn't take advantage of any OS facilities. Your new programs can. When you run a command program from PicoDOS8, you can specify arguments just as you would on a PC, and newer programs can access these from the argc and argv variables in the main() function. Similarly, when you chain to a new command program using the execstr() function, you can pass a command string that gets parsed into the argument variables.

BATCH Files

Also like DOS, PicoDOS8 allows you to automate some processes by running batch files, which are text files containing valid PicoDOS8 commands, and which have the filename extension .BAT. Our batch file processor does not support variables, looping or conditionals and there are no environment variables for the system to work with.

The file AUTOEXEC.BAT, if present, will load and execute immediately upon entering the PicoDOS8 command shell. This is a convenient way to automatically run an application at startup.

Trademarks: