PicoDOS8 & ANSI C Programming Guide

About the PicoDOS8 ANSI C Programming Guide

How to use the PicoDOS8 ANSI C Programming Guide

This guide is written as a follow-up to the printed Getting Started Guide that came with your Persistor and builds on information presented there. It, in turn, should have already convinced you of the necessity of mastering the TT8 using Onset's documentation. That done, you should be ready to start using the Persistor with C.

Other PicoDOS8 Related 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 any of the programming 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 User's Manual

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.

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

In the getting started guide we said it only took four lines of additional code to adapt a large disk intensive game program to use the Persistor? Well actually, only two of those lines relate to the PicoDOS8, the other two specified the database filename and did the standard InitTT8() call.

For the same reason it took only two lines to adapt the adventure program, it'll be the same two lines to adapt you to programming with the Persistor. Almost all of your interfacing will be through the standard C library function calls documented (among other places) in the ANSI C version of The C Programming Language by Kernighan and Ritchie. In fact, though we provide an API for accessing command line options and chaining to other applications, there are no other lower level file access interfaces provided with PicoDOS8.

#include <PicoDOS8.h>

This is where all the action takes place - most of which you can safely ignore. Near the top is an inline function and three C macros that you'll be using in the main() function of your PicoDOS8 aware applications.

#define CF8StdCS   2   // defaults to CS2

#define CF8StdAddr   0x800000 // gotta be somewhere

#define InitCF8(cs,ad) \...

The macro CF8StdCS just specifies the chip select used for the Persistor CF8. This is hard wired to CS2, but can be changed with a cut and jump as described in the Persistor Hardware Manual. The second macro, CF8StdAddr, simply tells the Persistor where it lives in memory. This location seemed as good as any, but you can move it around if it conflicts with your custom hardware.

If you're interested, the InitCF8 macro first invoke the inline function initpd(). If you were to disassemble this, you'd see that it first sniffs around in high flash where PicoDOS8 is known to hang out, and on finding it, knows where to go to have the TRAP12 handler installed. The next calls to initcf() and initfs() use the newly installed TRAP mechanism to do their respective jobs of initializing the Persistor and initializing the file system.

InitCF8(CF8StdCS, CF8StdAddr);

This is the line of code you need to add to your main() function to enable PicoDOS8 services. It must immediately follow your call to InitTT8(), which as you recall, must be the first executable statement in your main() function. Typically, your main() function will begin like this:

```
main()
  {
  // local variable declarations ...
  InitTT8(NO_WATCHDOG, TT8_TPU); // setup
    Model 8
  InitCF8(CF8StdCS, CF8StdAddr); // always
    follows the InitTT8() call
  if (errno != 0) // look out for No
    Hardware (100) or No Media (101)
  printf("\n\n!!! InitCF8 failed, error %d
    !!!\n\n", errno);
  if (errno == -1 || errno == PiDosNoHard-
    ware) // no PicoDOS8/Persistor !
  Reset(); // any future PicoDOS8 calls
    would crash the TT8
```

Smart programmers will check the global variable errno to make sure that initialization succeeded before launching into the application code, but barring some hardware problem, your Tattletale is now prepared to use the entire standard C library file functions. If the call to InitCF8 returns -1 (no PicoDOS8 found in high flash) or the constant PiDosNoHardware (defined in <PicoDCF8.h>), you must not attempt further PicoDOS8 calls since there is nothing at the receiving end to handle them and your program will crash with an exception error. If the call returns PiDosNoMedia, there's no card in the Persistor, and you'll be limited to functions that can operate without a card (like some of the execstr() commands).

argc = initargs(&argv);

Now that your program is PicoDOS8 aware, you may want to take advantage of the ability to receive, and even pass on, command line arguments. Since PicoDOS8 wants to coexist with non-PicoDOS and legacy applications, you must perform an additional step you don't normally see in PC hosted programs.

At the point of entry, argc will be equal to one and argv[0] will point to some harmless but meaningless short string. To get the command line values, add this initargs line before you access the argc or argv parameters (but after calling InitCF8()):

After that, you can treat the arguments just as you would in any C program. One of the standard PicoDOS8 example programs, called TestArgs, demonstrates how to use this capability. You can compile and run it directly, but it's most informative if you load it and save it to the card as a runnable (testargs.run) file. From the PicoDOS8 prompt, you can then type "testargs", followed by some parameters, and it will show you how it receives and interprets the parameters.

*short pdcfinfo(char *drive, long *size, long *avail);*

This call returns the size of the Flash memory card and the amount of free space remaining. Pass the string "A:" for the drive parameter, and pointers to longs to hold the size (max) of the card and the free space available. You should be aware that this call can take several hundred milliseconds before it returns.

```
long    cfsize, cffree;
pdcfinfo("A:", &cfsize, &cffree);
```

*int execstr(char *cmdline);*

To make up for the flash we take away to host the PicoDOS8 kernel, we provide the execstr() function to allow you to chain to other command programs on the Flash memory card. Usually, you can ignore the return value, since it's not supposed to return. If it does, the requested program never ran, and you can take a look at the returned error code for a clue as to why. The following C statement runs the TESTARGS.RUN program just to show off.

```
err = execstr("TESTARGS -p100 -f988
blah.bla");
```

Obviously you don't have to feed in string constants and could vector control from a lookup table or even direct or remote user input.

*char *picodosver(void);*

This call returns a string defining the PicoDOS8 target, serial number, and version number in the form "TTT#SSSS-V.RR", where TTT is the Persistor type (CF8), SSSS is the serial number, V is the PicoDOS8 version, and RR is the current release. Note that beta releases of PicoDOS8 may have a single character appended to the release number. This current version of PicoDOS8 for Persistor CF8 serial number 124 would be: "CF8#0124-1.33".

void ResetToPicoDOS (void);

This call simply jumps to PicoDOS8 (0x2BCF8) as a way to get back to PicoDOS8 from an application burned in flash. Be aware this could fail if you've installed custom interrupts.

Contacts:

Persistor Instruments Inc.

Tel:	508-759-6434
Fax:	508-759-6436
e-mail:	info@persistor.com

Onset Computer Corporation

Tel:	800-LOGGERS
Fax:	508-759-9100
e-mail:	sales@onsetcomp.com

Trademarks:

Persistor®, PicoDOS®, and MotoCross® are registered trademarks of Persistor Instruments Inc. **Tattletale** and **CrossCut** are trademarks of Onset Computer Corporation. **CompactFlash™** is a trademark of the CompactFlash Association.