

## **The New World of Flash Microcontrollers and the MSP430**

In recent years flash based microcontrollers have progressed from being a rare exception to being the norm. Up until now most of this new technology has been used as asked ROM substitute – so why introduce the additional expense? What can it do for my product?

Well imagine a microcontroller with 16K Flash 256 bytes RAM, a three wire serial connection and some data recording/playing interface. The device appears as a pen drive on your PC – you can edit an ini file on the target using notepad – you can drag and drop a firmware update to the drive – you can drag and drop new wav files to the target – you can open a diagnostic file on the target device and get dynamic updates.

These are just a snapshot of the possibilities opened up by the MSP430 and uCDrive software.

### **Why MSP430?**

The MSP430 is a unique microcontroller in many respects ..... But the most undervalued part of its architecture is its internal flash. It has a unique combination of design characteristics which make it particularly suited to developing sophisticated solutions. These key features include:

1. small erasable sectors
2. byte writable flash
3. low power flash
4. 100,000 erase/write cycles
5. wide selection of configurations. (1K-128K of flash)

### **Potential Applications**

There are many possibilities that this technology can unleash:

1. Firmware upgrades in the field

No longer must your microcontroller product be programmed leaving the factory and never change – using a safe bootstub standard interfaces can be used to completely reprogram your device

2. Configurability in the field

And all those microcontroller products that must have fixed configurations or switches and leds to configure with. Now you can just drop a new configuration file to it; set local variables and language files by a drag and drop on your PC etc. Edit a .ini file with notepad.

3. Diagnostic

Information recorded by the microcontroller can be written to a standard file which can be opened and read by a standard windows application. No more

special “engineering DOS tools” – use standard programs with standard interfaces.

#### 4. Data Logging (maybe enhanced with serial flash)

Create a data logger product with automatic access to the data. With the addition of a serial flash chip the storage capability of such a solution can be increased to 8Mytes or more.

### **Challenges**

To be able to provide these solutions has its problems – reliably managing flash is always an interesting business. There are a number of problems that have to be solved and the exact set of problems is always dependent on the precise flash device used. The main problems with flash storage in this environment are:

1. partial erase or partial write – when a page is erased or written how do you ensure that the data is not misinterpreted because of unexpected power loss.
2. wear-leveling –how to avoid over use of a particular area of the flash
3. cumulative write and erase times – flash access needs to be carefully timed
4. file system RAM and ROM footprint – needs to be minimal for microcontroller.
5. fail-safety. You cannot run a checkdisk on a micro – the system must always be intelligible.

### **Solutions**

To exploit this new technology and solve all the issues HCC provide a customized file system for the MSP430 together with the uCDrive windows driver which allows the connection of this file system to a PC through a cheap and simple three wire serial connection.

### **File System on a Micro?**

Using a file system on a microcontroller is a very efficient solution for providing the required functionality. EDFS-TINY has been hand-crafted for the MSP430 and has a ROM footprint of 0-8K and a RAM footprint typically <100bytes. Even a 1K flash MSP430 device can appear as drive on a PC with editable ini files!

By using files for storage the developer achieves several major benefits:

1. The data storage is position independent – no longer must the developer worry about where his data is – what to erase – what happens if I increase this structures size – he just uses file access. All else is hidden from him.
2. Media Independence – the file API ensures your application is not dependent on the particular flash device – when a different media is used upgrade your file system – not your application.
3. Portability – when new targets are used with even small differences I the flash architecture – the developer need not worry – just use the latest file

system. The code is independent of the storage mechanisms. The file system does it all for you.

4. Accessibility – having your data in a file immediately makes it accessible by other applications. The application only needs to be able to open and read the file to use the data.
5. Core Competence – by using a reliable, much used file system the developer can forget about issues relating to how his data is stored and focus on his core competence – the technology for which the customer makes his mark.

## **Conclusion**

By using the MSP430 combined with suitable flash management software the developer can create products with new functionality and connectivity both quickly and reliably. The development overhead is minimized such that with a development board you can be accessing your MSP430 applications' configuration files within hours. With this turnkey solution any number of imaginative solutions can be implemented immediately with minimal risk.