Design of Matchbox, the simple filing system for motes

David Gay
Version 1.0

August 21, 2003

1 Overview

The mote filing system is designed to provide a simple filing system for mote-based applications. Its design goals and functionality are covered in the matchbox document; this document describes Matchbox's implementation for the mica family of motes.

This implementation is tailored to the Atmel AT45DB041B flash chip used on the Micas. It should consider the following constraints of this particular flash chip:

- The flash is divided into sectors (mostly 128K, though some are smaller).
- Each sector is divided into pages, each page is 264 bytes long.
- Pages can only be written as a whole.
- Pages should be erased before being written (whatever "should" means).
- After 10'000 writes to a sector, all its pages should have been written at least once.

Note that these are different from "usual" flash rules. These typically say:

- Each page/byte/whatever can be written a limited number of times (e.g., 100'000)
- Erases are on a larger granularity than writes (e.g., 64k-erases, 528-byte writes)

2 Data structures

2.1 On flash

The 264-byte page is divided into 256 data bytes and 8 metadata bytes. The metadata stores the following information:

- per-page CRC (data bytes only) (16 bits)
- page-write counter (16 bits)
- last-byte-used in page (9 bits)
- next page pointer (a page number), or `IFS_EOF_BLOCK` for the last page
- root page marker (3 bits)

- metadata check byte (8 bits)

Pages are numbered from 0 to ¡file system capacity in pages¿-1. The expectation is that the file system will use some whole number of flash sectors. Pages `IFS_FIRST_PAGE` through `IFS_FIRST_PAGE + IFS_NUM_PAGES` are requested from the `ByteEEPROM` component for use by the filing system.

There are two types of pages: data pages (root page marker is 0), and first-meta-data-page pages (root page marker is `IFS_ROOT_MARKER`).

The metadata check byte detects inconsistent metadata. It is the bitwise complement of the sum of the three bytes that store the last-byte-used in page, the next page pointer and the root page marker.

Files and the metadata are both byte-streams. A byte-stream is a sequence of bytes, and is identified by its initial page number. The pages of a byte-stream are found by following the linked list formed by the next page pointers. Data ends at the first page where last-byte-used is not 256. Note that the linked list of pages may extend arbitrarily past the last page containing data - this means that space has been reserved for future expansion of the byte-stream.

The per-page CRC is always used for metadata, it is used for files if the `FS_CRC_FILES` constant (in `Matchbox.h`) is true.

Meta-data format (viewed as a byte-stream) : A header stores the meta-data version number, followed by an unsorted list of file entries. Each file entry is a 14-byte null-terminated string followed by the file's byte-stream's initial page number (2 bytes). The first page of the meta-data bytestream has the first-meta-data-page page-type. The other pages of the meta-data, and the file byte-stream pages have the data page type.

Note that their is no fixed *root* page: the initial page number of the meta-data is found by scanning the flash looking for the page with type first-meta-data-page which contains the highest meta-data version number. This avoids continuously writing the *root* page (a bad idea given the limited number of writes supported by flash). Scanning the flash to locate this page at boot time is acceptable in Matchbox as the flash is small (a few 100k) and page access relatively fast (approx. 1ms).

There is also no explicit listing of the free pages, this can also be reconstructed at boot time once the initial page of the meta-data is found.

2.2 In memory

We must keep the following information (each piece of information is followed by the name of the component which owns it):

- Meta-data root page.
- Meta-data version number.
- Free page bitmap.
- Last allocated page.
- Number of free pages.
- For each read file descriptor:
  - initial page number
  - current page
  - next page

  - current page offset
  - last offset on this page
  - use-crc-for-this-file flag

- For each write file descriptor:

  - initial page number
  - current page
  - next page
  - the current page's number (first page is 0, second is 1, etc)
  - current page offset
  - last offset on this page
  - remaining reserved bytes
  - use-crc-for-this-file flag

- various pieces of miscellaneous state for operations in progress

Matchbox uses one read and one write file descriptor internally (for the metadata).

3 Algorithms

This section mentions a few basic principles, rather than cover the (straightforward) algorithms for each operation:

Update of meta-data is always atomic: new meta-data is written by making a copy of the old meta-data in new pages, with appropriate changes. The new initial metadata page contains a higher version number than the current metadata. The initial page is only marked as a first-meta-data-page once all other metadata has been written. New meta-data is only written after the information it refers to is known to be valid (e.g., on new file creation, the file's first page has been successfully written).

At boot time, the initial metadata page is located by reading all of the flash sequentially and finding the valid (by reling both on the metadata check byte and the per-page CRC) first-meta-data-page page with the highest version number.

Once the initial metadata page is located, the free page bitmap is built by walking through the filing system. Inconsistencies in free pages will lead to the filing system being marked readonly. A separate fsck-style tool should be written to fix this (note that this should only happen as a result of page corruption given the atomic metadata updates, hence should be hopefully rare).

A corrupt file is one containg a page with a bad CRC. This could be due, e.g., to a power-off during a file-page write. Currently, attempts to read such a file will fail when the corrupt page is reached. Attempts to open a corrupt file for writing will fail.

A free page is found by scanning the free page bitmap starting at the value of *Last allocated page* (and updating *Last allocated page*). This produces some amount of "wear levelling" (i.e., writing each page an equal number of times). The boot-time value of *Last allocated page* is the meta-data initial page number.

A background task sequentially walks over the filing system's pages and checks their write counters. Any page which is more than 9000 writes "old" is rewritten (and hence sees its write counter updated). The write counter value is maintained inside the EEPROM component. A walk rate of one page per minute should be sufficient (assuming that write-rate is not high - may need further analysis and/or adaptivity).

4 Interaction with TinyOS and Current Status

Matchbox will interoperate with direct access to the flash chip via the `ByteEEPROM` component. Matchbox reserves its area of the flash chip with `ByteEEPROM` at boot time. It is possible to change this area by editing the `IFS_FIRST_PAGE` and `IFS_NUM_PAGES` constants in `tos/lib/FS/IFS.h`.

The write counters and the background task which rewrites pages have not yet been implemented.