A Simple Virtual FAT File System for Wear-Leveling Onboard NAND Flash Memory

Robert Klar, Sue Baldor, Charles Howard,
Randal Harmon, and Allison Bertrand
Southwest Research Institute

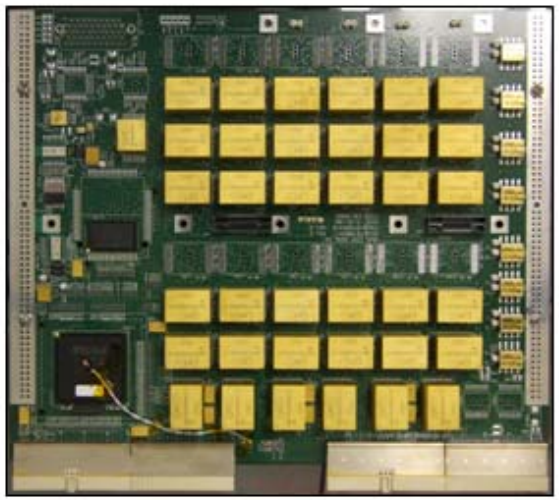
Promise of Flash

- Flash Memories originally developed for the consumer market have revolutionized mass storage for Space-based applications
  - High density parts have been screened and approved for use
- Flash provide many desirable features:
  - High memory density
  - High tolerance to radiation
  - High speed (as compared to older technologies such as EEPROM)
  - Low power requirements
  - Non-volatile

File Systems

- File Systems provide a method to logically organize data into discrete units
- File Systems provide a useful abstraction that helps to reduce system complexity and make efficient use of data storage
  - Many historical file systems were developed for magnetic disks since this was the prevalent mass storage technology for many decades
  - Since Flash technology has become widely available, some newer File Systems have been developed to support it
    - TrueFFS (formerly M-Systems)
    - YAFFS (Aleph One)
    - JFFS, JFFS2 (Linux Open Source Community)

Motivation: Mass Memory Board



EM Mass Memory Module (MMM)

- A high-capacity Mass Memory Board was designed for the Magnetospheric Mutiscale (MMS) Mission
  - 128 Gigabytes of Flash
  - 1 MB of SRAM and EEPROM
  - 6U cPCI Form Factor

- MMS Mass Memory was designed to support a hybrid CCSDS File-Delivery Protocol implementation (described at FSW-08)
- A smaller form factor board has been developed to support multiple mission applications
  - 48 Gigabytes of Flash
  - 1 MB of C-RAM
  - 3U cPCI Form Factor
- A newer 6U board is also in the works
- A lightweight file system is desirable to support a wider variety of storage applications

Software Goals/Requirements

- Make efficient use of the Mass Memory Board
- Provide wear-leveling of the Flash in order to maintain storage capacity for missions of a few years duration
- Provide support for file naming and directory structures
- Support common embedded operating systems such VxWorks and RTEMS
- Require only a small amount of processor memory
- Provide for a responsive system
- Provide software which can be quickly made available to customers and put to use
  - Both TrueFFS and YAFFS have additional third party licensing requirements for use in commercial products

Reason for Wear-Leveling

- NAND Flash is a limited-life component
  - NAND Flash is rated at ~100,000 erase cycles
    - Flight-qualified Flash is derated to ~10,000 erase cycles
- Traditional File Allocation Table (FAT) File Systems were not designed for use with Flash devices
  - FAT is stored in a fixed physical location resulting in frequent updates to the same physical erase block
    - A conventional FAT file system would exceed limit very quickly without wear-leveling
    - For example, for an 8MB file, using a Cluster size of 8K, the erase limit would be exceeded on the FAT after only 10 writes of the file

      8 MB / 8K = 1024 writes to the FAT per file write

      10 * 1024 = 10240 writes to FAT for 10 writes to file

Where to do Wear-Leveling

- Wear-leveling can be accomplished at several levels:
  - Hardware Controller
    - Off-the-shelf USB Flash Drives commonly use this approach. This allows the use of file systems not specifically designed for Flash.
  - Device Driver
    - This offers some of the same benefits as a hardware controller but adds a little complexity to the device driver. It has the advantage that we can make use of some of the features provided by the hardware design.
    - This is the alternative we chose!
  - File System
    - This also works well but may bring along some additional overhead. Configuring a file system to make use of your particular Flash device requires some work.

Structure of FAT File System

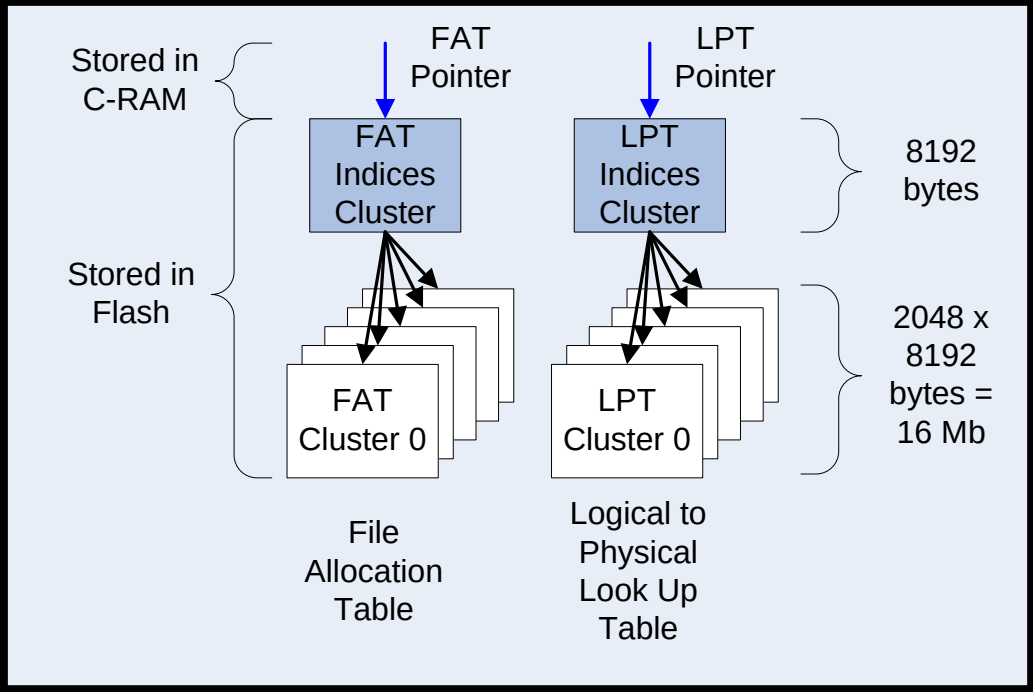
- Common FAT File Systems are FAT12, FAT16, and FAT32
  - FAT32 is the newest of these and includes support for the largest number of Clusters
  - FAT12 and FAT16 are not suitable for large capacities
- Smallest organizational unit is a Sector = 512 bytes
- First sector on media is the Boot Sector
  - This sector contains information about how the media is formatted
  - The Boot Sector includes the number of Sectors per Cluster
- Following the Boot Sector are the File Allocation Tables
  - Allocation unit is a Cluster
- Following the File Allocation Tables is the Root Directory
  - Root Directory can be located anywhere in data area in FAT32

Approach to Wear-Leveling

- Use conventional FAT32 File System but relocate the FAT
- Do Logical to Physical Mapping
  - As FAT is updated, move portions of the table in order to evenly distribute erase cycles
- Do Garbage Collection
  - Reclaim blocks when majority of pages are no longer used
- Take advantage of hardware features
  - Partial page writes: a Flash page may be partially written up to 4 times with each erase cycle
  - Chalcogenide RAM: A small amount is available on the board. C-RAM is faster and provides smaller addressable units. However, it is not as dense as Flash.
  - Metrics: Hardware keeps track of erase status, number of erases and bad block vector.

Data Structures

- FAT and LPT Pointers (C-RAM)
- File Allocation Table (Flash)
  - Relocated to Data Area Blocks
- Logical to Physical Look Up Table (Flash)
  - Map Logical Cluster to Physical Cluster
- Cluster Selection Heap/Map (RAM)
  - Prioritize Partially Empty Erase Blocks by Number of Erases



- On the Mass Memory Board, we use 16 Sectors per Cluster = 8192 bytes
  - This corresponds to a Write Page
  - An Erase Block is 512 kilobytes

Data Structures

- A group of Cluster size blocks of C-RAM are reserved for the FAT and LPT Pointers
  - As these pointers are updated a new word is used to store it
  - This provides for wear-leveling of C-RAM
- Clusters can be in one of several states in Cluster Selection Map
  - USED, INVALID, or ERASED
  - INVALID Clusters are reclaimed by erasing the corresponding block and rewriting the USED clusters in it. Except for this additional step, the Cluster Selection Algorithm treats ERASED and INVALID Clusters much the same.

Writing to Flash

- Operating System invokes driver routine to write a Logical Cluster
  - This may be a Data Cluster or a FAT Cluster
- Driver looks up in LPT to find current Physical Cluster corresponding to the Logical Cluster
- If Cluster is USED, then the Driver marks it as INVALID in the Cluster Selection Map
- Driver selects Physical Cluster using Cluster Selection algorithm
- Driver writes to Physical Cluster
- Driver updates LPT
- Driver marks Cluster as USED in Cluster Selection Map

- Cluster selection algorithm
  - A heap is used to prioritize Erase Blocks which contain INVALID or ERASED Clusters
  - Next Physical Cluster is chosen to be in the Erase Block with the fewest number of Erase Cycles

Garbage Collection

- Some Erase Blocks will contain data that seldom changes. Over time there will be a disparity in the number of erase cycles between these blocks and others.
  - To avoid this, Erase Blocks which contain only USED Clusters but that have few erase cycles will be periodically moved so that these become available to the Cluster Selection Algorithm

Worst Case Performance

- For each Data Cluster write there are additional writes to update data structures
  - 1 write to corresponding FAT Cluster
  - 1 write to FAT Indices Cluster
  - 1 write to corresponding LPT Cluster
  - 1 write to LPT Indices Cluster
- Total of 4 Cluster writes to Flash
  - 1 write to FAT Pointer
  - 1 write to LPT Pointer
- Total of 2 writes to C-RAM

Potential Optimizations

- If 1's change to 0's in FAT or LPT, then write page up to 4 times before moving it
- Use disk caching to avoid repeated updates to Clusters in Flash

- Experimental Prototype in Development
  - Validation Testing planned to completed by early next year

Questions?