

# Hard disks are dead. Now what?

Jörn Engel  
<joern@logfs.org>

## Abstract

For some years hard disks have been the preferred storage medium in most people's computers. They were cheap enough, large enough, fast enough and robust enough - or at least people learned to perceive them as such. Compared to various alternatives, they certainly had the right combination of check, large, fast and robust.

But a new alternative is emerging that, while still smaller and more expensive, is significantly faster and more robust: flash. And if current trends continue, flash will shorten the price gap and become the preferred storage medium of the future.

So how what the brave new storage world look like? This paper will take a look at current flash storage and show some of the differences to hard disks and possible problem areas.

## 1 Flash introduction

Many types of flash exist, but when talking about replacing hard disks as mass storage media, one almost always talks about NAND flash. So allow me to ignore all other variants of flash in the remainder of this document.

Hard disks are organised in sectors that can be read and written in any order. Not all orders yield the same performance, though. Hard disks work particularly well when accessing data sequentially, able to sustain tens or even hundreds of megabytes per second. Truly random accesses are slow and only about 100 can be performed per second.

Flash is organised in pages and blocks. Blocks can be read, written and erased in any order, so long as a block is always erased before being written. Individual pages that compose a block must be written in sequential order. Pages from different block can be written in any order. It is possible to write one page from one block, wait for years

and write thousands of pages in different blocks, then write the next page from the original block.

Like hard disks flash contains additional storage for metadata that is private to the device or its driver. Ordinarily a 512 byte page contains 16 bytes of out-of-band (OOB) information.

Unlike hard disks, the chances of individual blocks to fail increases with the number of erases. Manufacturers usually guarantee either 10,000 or 100,000 erases for their chips. In order not to exceed this number, a technique called Wear Leveling is usually employed.

## 2 FTL

Linux has an entire subsystem devoted to handling flash, the Memory Technology Devices or MTD subsystem. Most users of flash however haven't ever used or even heard of this. Most flash available to the consumer, be it USB-sticks, CompactFlash, Memory Sticks, MMC, SD, SmartMedia or xD are accessed as block devices, just like disks.

We have seen in 1 that flash is indeed quite different to disks, so something must be done to make it behave reasonably similar. That something is commonly known as a Flash Translation Layer, short FTL. It is a layer on top of flash that allows sectors to be read and written in any order without knowing what a page or block is or how to deal with them.

At the time of this writing, the Linux kernel supports five different variants of an FTL in the MTD subsystem alone. In addition, a number of USB mass storage drivers contain their own, appropriate to their supported devices. However, one of them appears to beat all others in terms of sold devices. That one is SmartMedia.

## 3 SmartMedia

SmartMedia used to be a common form of consumer flash device, these days limited to the second hand market. What made SmartMedia special appears to be a very simple yet effective FTL and in particular the fact that it was reasonably well-documented in a market full of trade secrets. It can even be downloaded [1] by individuals.

Alternate descriptions [2] exist as well. So far I have encountered two individuals that independently reverse-engineered the SmartMedia format. One reason for this may be ignorance, another the fact that SmartMedia documentation is copyrighted and explicitly not allowed to be reproduced.

A third reason is that SmartMedia format is used far beyond the SmartMedia devices themselves where people would expect it. So for the remainder of this document, the term SmartMedia will refer to the format, independently of what devices it may be found in.

## 4 How it works

In essence, SmartMedia maps logical addresses to physical ones on a block granularity. Early versions mapped the complete chip, later chips with growing capacity caused the mapping to be done in groups of usually 1024 blocks, called Zones.

Each zone consists of 1024 physical blocks that get mapped to 1000 logical ones. That leaves 24 blocks spare for various uses. One use is handling bad blocks. Small production defects can be handled by simply not mapping the affected blocks. But the main reason for spare blocks is the write behaviour.

When writing a sector, SmartMedia cannot just erase a block and write the new sector to it. You may remember that writing to flash requires erasing the block first, thereby losing all other data within it. So instead, SmartMedia writes to one of the spare blocks instead.

In the simplest form, it copies all data before the requested sector from the old block to the new one, then writes the sector itself, followed by old data behind the requested sector. If all goes well it then erases the old block so that only one physical block is mapped to the logical address.

If things don't go well, it is possible for the system to crash or lose power while writing the new block, erasing the old block or right between. In such a case, two block with the same logical are mapped to one logical address. When such a case is detected, it is necessary to decide which physical block is good and should be used.

Deciding on a good block is done simply by looking at the last page of it. Each page of a block contains the block's logical address in the OOB area. If the last page contains the same logical address, the block is assumed to be written completely and hence being good. If two good blocks exist, either one may be taken. The other one should get erased and made a spare.

Finding the correct physical block for a logical address requires reading the addresses of all block in a zone and potentially solving disputed with above method. Naturally this effort is only made once to create a Logical-Physical Conversion Table for this zone. Afterwards a simple table lookup is done.

## 5 OOB details

Byte	Purpose	Byte	Purpose
0	reserved	8	ECC 1
1		9	
2		10	
3		11	block address 2
4	12	data status	
5	block status	13	ECC 2
6	block address 1	14	
7		15	

Table 1: OOB usage

The following assumes 512 byte pages with 16 bytes of OOB per page. Adjusting it to other page sizes is left as an exercise for the reader. Table 1 outlines the OOB usage of SmartMedia. Byte 4 is used to indicated invalid data. Byte 5 indicates a bad block if two or more bits are cleared - after erasing, all bits are set.

Two fields contain the block address. Having two copies allows to catch bit errors. Another two fields contain ECC information for the page, covering 256 Bytes each. OOB data is always written after the 512 Bytes or in-band data are written.

A prudent reader may notice that the last Byte of OOB is not a block address field. This can be significant if the system crashes after writing the second block address, but before writing the ECC field. In

such a case the ECC will indicate random errors, possibly causing a "correction" that actually damages correct data. It might be a good idea if the ECC field was also checked in case of two block with the same mapping.

## 6 Performance tweaks

4 outlined the simplest possible form of writing to SmartMedia, which is also the slowest. Copying data from the old block can often be avoided. If the device receives a larger write request, several sectors can be written at once, possibly whole blocks.

Some implementations also wait after a write request, leaving the current block unfinished. If further data for the same block arrives, it is appended. Otherwise the old method is used, either after some delay or when writing to a different address.

A side-effect of waiting is that the current block would be considered invalid if the system crashed. That can have interesting consequences to databases calling `fsync()` and trusting the storage media to safeguard the data afterwards. Whether all implementations of such an optimisations are sync-safe is unknown and hard to test. As always, manufacturers tend to be secretive and consumers are left to hope for the best.

## 7 Wear Leveling

Flash can wear out if too many erases occur for the same block. Hence it is prudent to use the SmartMedia logical/physical mapping in a way that logical addresses get mapped to different physical block over time. This is called wear leveling. You author likes to stress that wear leveling is not actually the goal - avoiding wear-out is. The difference is subtle but in some cases it can matter a great deal.

With 24 spare block per zone, wear leveling is achieved simply by taking a new spare block whenever writing to SmartMedia. This strategy works surprisingly well for many use cases, but comes with two inherent problems. The first and easier problem arises from static data.

Static data is data that remains static - it is never changed. Many people therefore distinguish static wear leveling and dynamic wear lev-

eling. Dynamic wear leveling will only move changed data to different physical blocks, while static wear leveling will also move static data, at least once in a while. When such static wear leveling happens a write may take longer due to the additional data movement.

The second problem can be described as local vs. global wear leveling. SmartMedia only does local wear leveling - writes within a zone are spread roughly equally on all block within that zone. Blocks outside the zone do not participate in wear leveling. Global wear leveling would involve all physical block, independently of the zone containing them.

A simple way to achieve global wear leveling is to use just a single zone for all blocks. In fact, devices up to 16MB in size usually did that. However, as the device size and number of blocks goes up, the time required to build up the Logical-Physical Conversion Table also goes up, linear to the number of blocks. Long waiting periods before the first data can be accessed tend to make users unhappy, so devices of 32MB and more usually have several zones.

## 8 Possible improvements

Writing complete blocks and requiring global wear leveling are the two main problems of current approaches. And they are fundamentally hard problems. Efficiently serving small writes would require a mapping between logical addresses and physical data in granularities smaller than an block.

Going down that path requires quite a bit of infrastructure, namely a log-structured approach of writing to blocks complete with cleaner. Having such a stack of software in the small card for a digital camera does not seem very likely to happen.

So realistically a log-structured approach can only be implemented in the operating system, either as part of the storage subsystem or in a filesystem. And once this infrastructure is in place, it will not only work better with an FTL but obsolete large parts of the FTL functionality.

Probably the best improvement manufacturers can implement is to provide a method to bypass the FTL and allow raw access to the flash.

## 9 Patents

Many of the strategies outlined in this paper have been patented. Whether any of the patents filed would stand up in court is an open question. Which of the strategies exactly are patented and do not have prior art is another question best left for legal councils to answer.

This paper does not aim to give legal advice beyond the simple fact that many companies may claim right to some of the techniques described.

## 10 Conclusion

Having taken a look at the current state, our subject question remains unanswered. It is currently not clear what a world without hard disks would or should look like. However, this paper has given an overview of the current situation for consumer flashes and pointed out the problem spots. It's main purpose is not to document existing good solution but rather the lack thereof and the need for more work in the area.

## References

- [1] SmartMedia Web-Online Specification  
[http://www.ssfdc.or.jp/spec/index\\_e.htm](http://www.ssfdc.or.jp/spec/index_e.htm)
- [2] XD Media Specification (from Alauda project)  
<http://alauda.sourceforge.net/wikka.php?wakka=XdMedia>