swissbit®

Application Note

AN2104en

FTL types: block based vs. page based mapping

© Swissbit AG 2022
Creative Commons License[1]

[1]This work is licensed under the Creative Commons License "Attribution 4.0 International". To view a copy of this license, visit http://creativecommons.org/licenses/by/4.0/

swissbit®

Contents

1 Abstract

The flash translation layer (FTL) is the part of the firmware of a flash memory device that translates from the logical addresses of the host to the physical addresses of the flash.

This document describes the difference between FTL architectures with block based mapping and page based mapping, and the consequent impact on read/write speed and the write amplification factor (WAF).

2 Basics

The development of NAND flash is closely linked to the introduction of the digital camera in the mid-1990s. The first controllers used for flash management were commercially available 8-bit types with low processing power and low RAM. The complexity of the FTL had to be correspondingly low, but this was not a problem for the application case – the sequential writing of a few hundred kilobytes per photo. From this the block-based mapping developed, which became the industry standard for a long time, and is can still be found today with SD and CF cards as well as USB memories.

A NAND flash is made up of blocks. A block is the smallest erasable unit and consists of pages. It has a size of 2–20 MiB today, depending on the technology. A page is the smallest programmable unit and has a typical size of 16 KiB for current flash technologies.

The storage medium is addressed by the host using the so-called "logical block addressing" (LBA). The entire address space is divided into sections of 512 bytes, which are counted starting from zero in ascending order. The host can thus address a minimum of 512 bytes per read or write access. This logical block addressing is not related to the flash blocks.

3 Block based mapping

With block-based mapping, each flash block contains sequential ascending addresses. The FTL only has to manage the start address of each block.

As an example, consider a flash with the following properties:

Capacity :	16 GiB
Block size:	4 MiB
Page size :	16 KiB

This results in the number of flash blocks and the number of pages per block:

Blocks total :	$\frac{\text{capacity}}{\text{block size}} = 4096$
Pages per block:	$\frac{\text{block size}}{\text{page size}} = 256$

The FTL has to manage only 4096 entries in its table to find the block containing the requested address. Since the addresses within the block are sequentially ascending, the correct page can be addressed directly. The size of the table for 16 GiB claims only about 8 KiB RAM.

Block based mapping is now considered obsolete, as it is only suitable for read-only applications and sequential writing. For random

writing, this technique is inappropriate because changing the content of a single memory address requires copying the entire block, since only entire blocks can be deleted: For example, if the content of page #123 from block #444 is modified, pages #0 through #122 are copied to a new block, then the new content of page #123 is written to the new block, and then the pages #124 to #255 are copied to the new block. Then block #444 is deleted.

Random write access in block based mapping is therefore very slow and produces a high WAF – and thus a high wear-out.

4 Page based mapping

In page based mapping, only the addresses within a page are sequentially ascending. Accordingly, the administration effort increases to 4096 blocks ∗ 256 pages = 1,048,576 entries. The administrative effort thus increases significantly. For random write accesses also the speed increases and the WAF drops, because now the unmodified pages of the affected blocks must be copied no longer.

The page based mapping has not found much popularity, and is now barely found because it was quickly replaced by the sub page based mapping.

5 Sub page based mapping

While a management unit is still an entire page in the case of page based mapping, the sub page based mapping typically divides the entire flash into units of 4 KiB. The administrative effort increases again, but random writes of the host operating system can be optimally processed: Common cluster sizes used by file systems are also 4 KiB in size (or larger). If, for example, a small data record is attached to a file, then in a FAT file system often only two write accesses are necessary:

- The actual write access with the payload to the end of the file
- The update of the file size in the folder

As mentioned earlier, the smallest programming unit is a page. However, due to the logical division into 4 KiB sections, both write accesses can now be stored in a single page, so that only 50 % of the page remain unused unless further write accesses of the host take place in a timely manner. The WAF is therefore 2. With pure page based mapping, two pages would have to be programmed for the two write accesses of the FAT file system example; the WAF would be 4. In block based mapping, the WAF would be 512.

6 Comparison

Sub page based mapping provides fast random write access with very low WAF. The administrative effort compared to the block based mapping is huge. A fast controller with a lot of RAM is needed. Just holding the management table for 16 GiB flash needs at least 12 MiB memory.

Small controllers without external DRAM, such as those used for SD cards, USB sticks and CF cards, can no longer hold this amount in RAM, but instead have to reload the mapping data from the flash if necessary. As a result, random read access on these drives is significantly slower than with a block based FTL. The vast majority of applications, however, benefit from the fast random write access and the extremely low WAF. For applications with sequential writes and many random read accesses, it may be necessary to use an old block based mapping firmware architecture.

For storage media with SATA or PCIe interface, only high-performance (and expensive) controllers are used at high flash capacities, with full management data held in (D)RAM.

7 Summary

Table 1 shows the suitability of the various flash translation layers for the four access modes. While block based mapping is now used only for read-only applications, sub page based

Table 1: Flash Translation Layer

		sequential read	random read	sequential write	random write
Block based	Performance:	++	++	++	−−
	WAF:			+[a]	−−
Page based	Performance:	++	+/++[b]	++	o
	WAF:			++	o
Sub page based	Performance:	++	o/++[b]	++	++
	WAF:			++	++

[a] If written sequentially by means of a file system, some random accesses result from updating of the file system management.
[b] Only controllers that can hold the management tables in RAM perform very well.

mapping with its adaptation to the file system management sizes is the best solution for almost any application, especially as the low WAF often allows the use cheaper flash (MLC or pSLC instead of real SLC).

CONTACT US

Headquarters	**Swissbit AG** Industriestrasse 4 9552 Bronschhofen Switzerland	Tel. +41 71 913 03 03 sales@swissbit.com
Germany (Berlin)	**Swissbit Germany AG** Bitterfelder Strasse 22 12681 Berlin Germany	Tel. +49 30 936 954 0 sales@swissbit.com
Germany (Munich)	**Swissbit Germany AG** Leuchtenbergring 3 81677 Munich Germany	Tel. +49 30 936 954 400 sales@swissbit.com
North and South America	**Swissbit NA Inc.** 238 Littleton Road, Suite 202B Westford, MA 01886 USA	Tel. +1 978-490-3252 salesna@swissbit.com
Japan	**Swissbit Japan Co., Ltd.** CONCIERIA Tower West 2F 6-20-7 Nishishinjuku Shinjuku City, Tokyo 160-0023 Japan	Tel. +81 3 6258 0521 sales-japan@swissbit.com
Taiwan	**Swissbit Taiwan** 3F., No. 501, Sec.2, Tiding Blvd. Neihu District, Taipei City 114 Taiwan, R.O.C.	Tel. +886 912 059 197 salesasia@swissbit.com
China	**Swissbit China**	Tel. +886 958 922 333 salesasia@swissbit.com

Disclaimer:
The information in this document is subject to change without notice. Swissbit AG ("SWISSBIT") assumes no responsibility for any errors or omissions that may appear in this document, and disclaims responsibility for any consequences resulting from the use of the information set forth herein. SWISSBIT makes no commitments to update or to keep current information contained in this document. The products listed in this document are not suitable for use in applications such as, but not limited to, aircraft control systems, aerospace equipment, submarine cables, nuclear reactor control systems and life support systems. Moreover, SWISSBIT does not recommend or approve the use of any of its products in life support devices or systems or in any application where failure could result in injury or death. If a customer wishes to use SWISSBIT products in applications not intended by SWISSBIT, said customer must contact an authorized SWISSBIT representative to determine SWISSBIT willingness to support a given application. The information set forth in this document does not convey any license under the copyrights, patent rights, trademarks or other intellectual property rights claimed and owned by SWISSBIT.

ALL PRODUCTS SOLD BY SWISSBIT ARE COVERED BY THE PROVISIONS APPEARING IN SWISSBIT'S TERMS AND CONDITIONS OF SALE ONLY, INCLUDING THE LIMITATIONS OF LIABILITY, WARRANTY AND INFRINGEMENT PROVISIONS. SWISSBIT MAKES NO WARRANTIES OF ANY KIND, EXPRESS, STATUTORY, IMPLIED OR OTHERWISE, REGARDING INFORMATION SET FORTH HEREIN OR REGARDING THE FREEDOM OF THE DESCRIBED PRODUCTS FROM INTELLECTUAL PROPERTY INFRINGEMENT, AND EXPRESSLY DISCLAIMS ANY SUCH WARRANTIES INCLUDING WITHOUT LIMITATION ANY EXPRESS, STATUTORY OR IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

© 2022 SWISSBIT AG