swissbit®

Design-In Guide for Swissbit Flash

Application Note

BU: Flash Products
Date: 9 November 2012
Revision: 5

swissbit®

Content

Release history:

Revision	Changes	Date
1	– First release	August 2, 2010
2	– USB status tools replaced by SBLTM	October 18, 2010
3	– CompactFlash cable length specification – Added a paragraph master/slave configuration Windows – Added note about Indexing service in paragraph “Disable drive indexing” – Additional explanations in paragraph “Disk write cache policy” – Added comparison graph between aligned and not aligned filesystems	June 8, 2011
4	– Restructured Life Time Monitoring information – Updated Linux tools descriptions – Added information about TRIM and NCQ features	September 21, 2012

1 Overview

This document provides a guideline to optimal design-in of Swissbit NAND Flash based devices into customer systems.

The main goals for optimal use in general are:

- Optimal performance
- Optimal lifetime
- Optimal power fail stability

There are a number of influences on the operation of Swissbit Flash devices, which will be shown in this application note. We will also discuss tricks and tips that can be implemented with low effort, but will result in big advantages.

Most information applies to all product types using Flash memory (CompactFlash, CFast, SSD's, USB Flash Devices, SD, MMC, microSD, etc.), but some are specific to the interface (like IDE, SATA, USB, ...) of the products. Information only applying to a certain product group is marked as such.

Other information is specific to certain operating systems, this also marked explicitly.

1.1 Technical documents, references

Please consult the datasheet of the product in question for more technical details and properties of the product type used.

1.2 Swissbit support

Of course Swissbit sales and technical support is available for competent and fast support. We will be glad to answer your questions or discuss system specific questions.

Please contact us via your established sales contacts. Contact information is also available on our website [1].

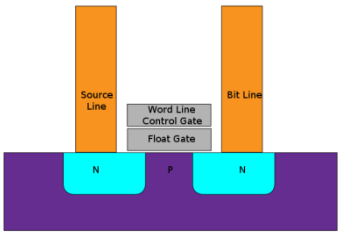
[1] http://www.swissbit.com/

2 Flash devices: internal operation

Please see our application note "NAND Flash based Solid State Storage" for a more detailed description of NAND Flash memory. This application note will only cover basic functionality which helps understanding the following chapters.

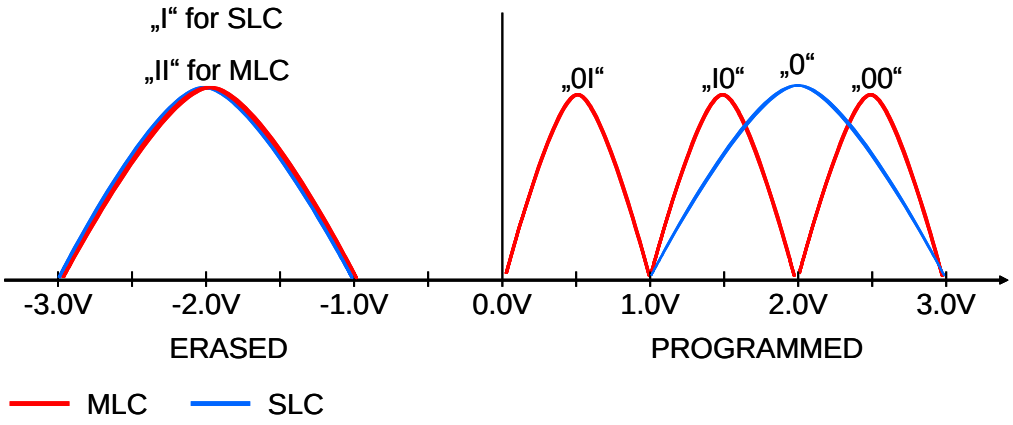
2.1 Principles of operation

Flash memory stores information in an array of memory cells made from floating-gate transistors. In single-level cell (SLC) devices, each cell stores only one bit of information. Some newer Flash memory, known as multi-level cell (MLC or TLC) devices, store more than one bit per cell by using multiple levels of electrical charge to apply to the floating gates of its cells.



Flash memory cell

Writing and erasing imposes stress on the individual cell so that cell ages or wear down over program/erase cycles. In effect, the charge levels change over time. While programming time decreases, erasing takes an increasing amount of time. Eventually charge levels exceed the defined thresholds and errors occur. As thresholds for MLC must be closer together, production variances have a greater effect and errors occur more readily and/or earlier over time. Because of these effects, the number of erase cycles per Flash block is only guaranteed up to typical 100'000 for SLC Flash, and typical 3'000 to 10'000 for MLC Flash.



Swissbit only uses high quality SLC NAND Flash in its industrial products.

2.2 Flash Memory Organization

When looking at how a Flash memory is organized the smallest logical/administrative unit is a sector. Each sector contains 512 bytes plus an overhead area (traditionally 16 bytes). One to eight sectors are then grouped into pages, in the range of 512 to 4,096 bytes per page. At the next level of hierarchy blocks can include 32 pages of 512 bytes for example, or more recently 64 pages of 2,048 bytes. Blocks therefore also have a defined number of sectors, currently between 16 and 512, and there are 1024 to 8192 blocks per Flash device.

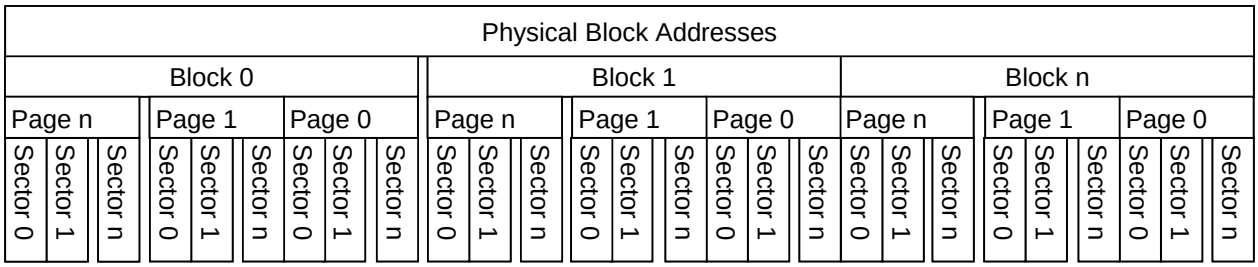


Figure 1: Organization of a Flash Memory

While Flash memory can be written / programmed at page level, it must be erased at block level. Erasing generally sets all bits in the block to 1. Starting with a freshly erased block, any location within that block can be programmed. However, once a bit has been set to 0, only by erasing the entire block can it be changed back to 1. Depending on Flash memory type, pages can be programmed in parts, up to 4 times (typically 4 for SLC Flash, 1 for MLC Flash).

As blocks are the "management" or "administrative" units, blocks will wear out as memory cells break down after a number of erase cycles. When defective, these blocks will be considered "bad blocks".

Typical SLC NAND Flash guarantees a life time of 100'000 erase cycles per block. Please consult the product datasheet for detailed information.

2.3 Flash Memory Controller and Firmware

The complex nature of Flash cells and their organization demands reliable, high performance control functionality. Flash Controllers of all kinds consist of an interface to the Flash memory, a processor and a host interface.

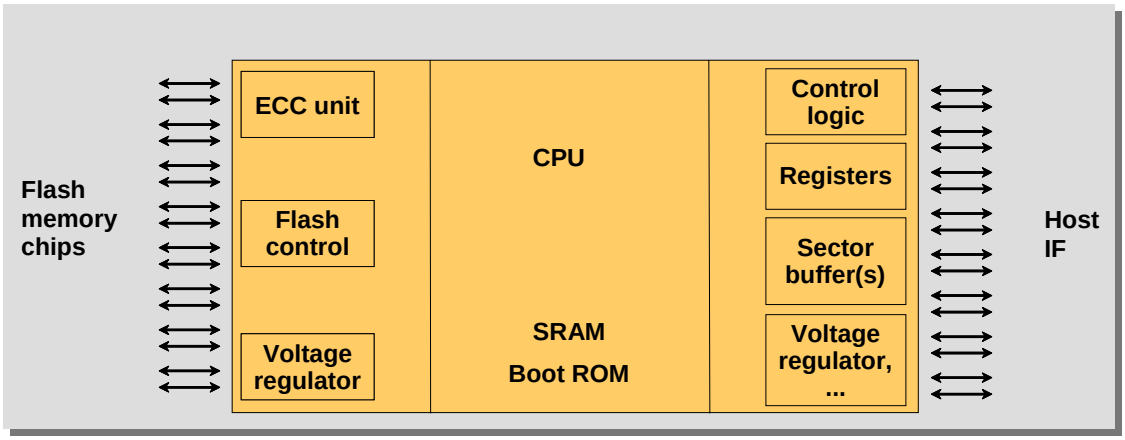


Figure 2: Generic Block Diagram Solid State Memory Solution

The controllers are based on a CPU together with dedicated hardware blocks, including an error correcting code (ECC) unit, buffers, Flash and host interface control logic.

2.4 Flash Memory Management

Several algorithms and concepts are used to address the questions initially posed in re-writing to areas already containing data, maximizing Flash life time, and ensuring data transfer integrity.

2.4.1 Blocks and Block Mapping

Logical block addresses (LBA) including the related static sector address or information are mapped correspondingly to physical block addresses (PBA). A table is maintained by the controller or firmware, translating requests for the particular LBA to its corresponding PBA. Logical blocks are distributed across all available Flash memory components and 'good' blocks, e.g. logical block 0 might correspond to a physical block at memory component 1 block 2, and logical block 1 might correspond to a physical block on memory component 3, block 3.

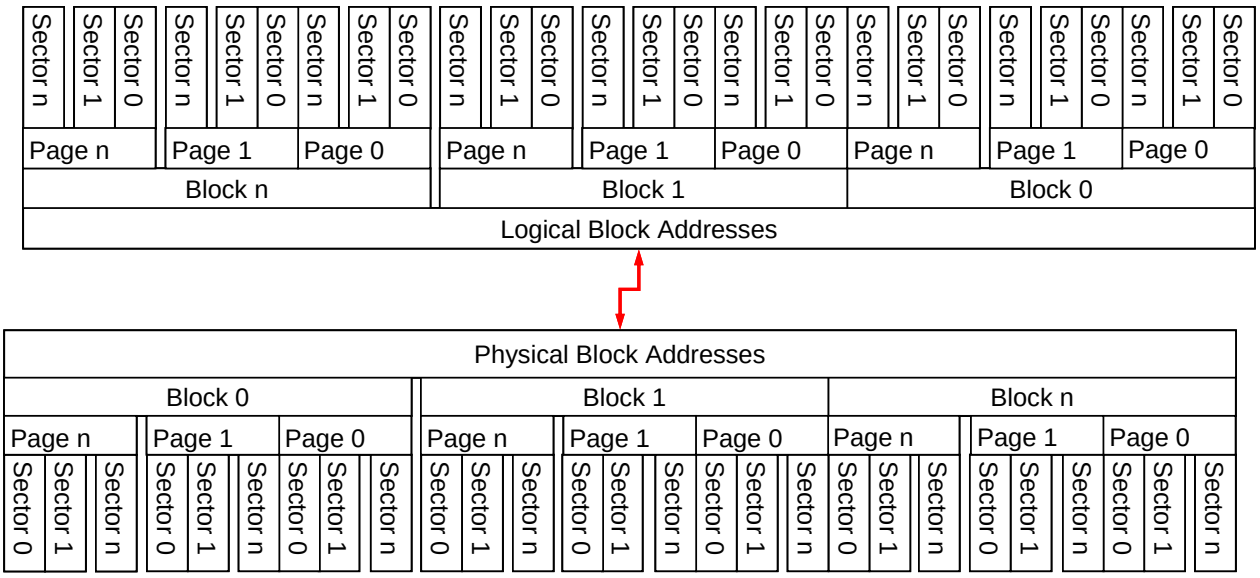


Figure 3: Logical to Physical Block Addressing

2.4.2 Error Code Correction (ECC)

The ECC is generally responsible for ensuring the quality of data being read, by adding information during the writing phase to restore partially corrupted data. ECC is an integral part of NAND Flash technology; all Flash controllers must implement ECC error correction according to the requirements of the NAND Flash. There are different ECC algorithm implementations used in Swissbit products, depending on the product group, please consult the datasheet for details.

2.4.3 Bad Block Management

In order to ensure that all blocks used for data storage are functional, but knowing that blocks wear out or become defective for several reasons, bad block management is necessary to retire all questionable blocks. During initial pre-formatting, Flash memories are tested and bad blocks originally marked by the Flash manufacturers are subsequently mapped out. Bad blocks are entered into a bad block table referring to physical block addresses. A pool of spare blocks is defined and used for dynamic bad block replacement, where blocks from the pool are used to replace blocks that produce errors when erased or programmed. This is called 'bad block re-mapping'. Considerations as to when blocks should be considered "bad" range from recognizing write or erase fails, reported by the Flash itself, to algorithms based on bit errors recognized by the ECC during reading. During the Flash's life within an application and while managed by an external controller, the bad block table is maintained and extended by any block going bad as it is mapped out or 'retired'. As soon as all spare blocks are consumed, the device is set into read only mode.

2.4.4 Wear Leveling

The wear leveling algorithms balance the use of all blocks, thus guaranteeing maximum lifetime of overall product. Even if only small parts (e.g. one file) of a device is written regularly, wear leveling will make sure all Flash blocks are evenly used.

There are different implementations used for wear leveling, also known as "dynamic" and "static" wear leveling. Dynamic wear leveling only rotates buffer blocks and the blocks that are currently written to – static data is not wear leveled. Static wear leveling also includes static data into the wear leveling, and thus results in much better endurance. All current Swissbit Flash based products use a combination of dynamic and static wear leveling.

Many products do wear leveling on a part of the device (per Flash memory component). Newer controllers also support "global wear leveling", which will provide wear leveling over the whole device.

2.4.5 Power loss protection

Flash memory is often used in removable storage applications or battery operated devices where a robust and reliable power source cannot be guaranteed. A user may remove the memory at any time and under these conditions security of data is of paramount importance. All current Swissbit Flash storage devices use a concept in order to ensure data integrity when transferring or writing data.

Upon a sudden power fail, the controller is reset and the Flash is immediately write-protected. The exact power fail algorithms used by the controller depend on the product group.

Swissbit performs extensive power cycling tests to all products verifying no data corruption due to power failure.

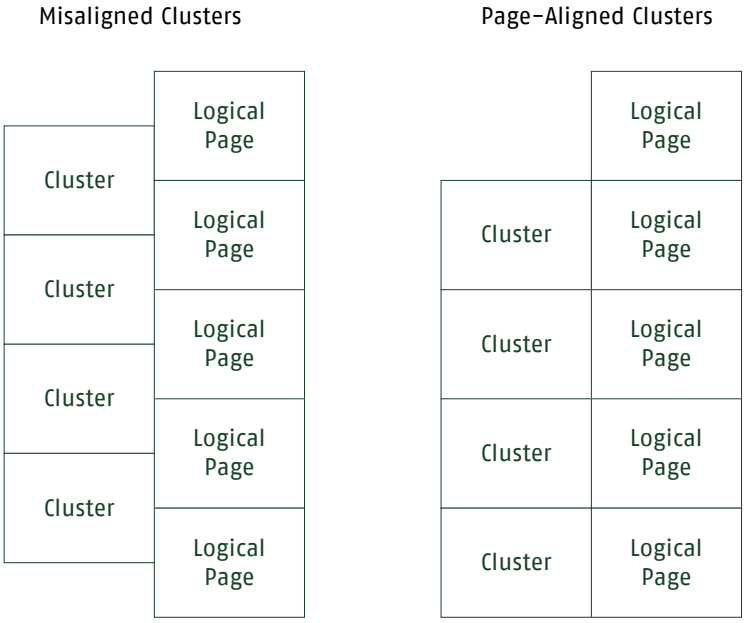
2.4.6 Logical page sizes

For better performance, modern controllers and Flash support various other features: multiple parallel Flash channels, sometimes interleave on the individual Flash channels (parallel access on multiple Flash memory components on the same channel), and Flash features such as "2-plane commands" or "copy back".

These performance optimizations have an influence on the logical page sizes: a factor of 1 up to 16 on the Flash page size applies. The exact factor depends on the exact model and capacity. Currently a logical page can be assumed to be max. 64kB.

2.5 Conclusions

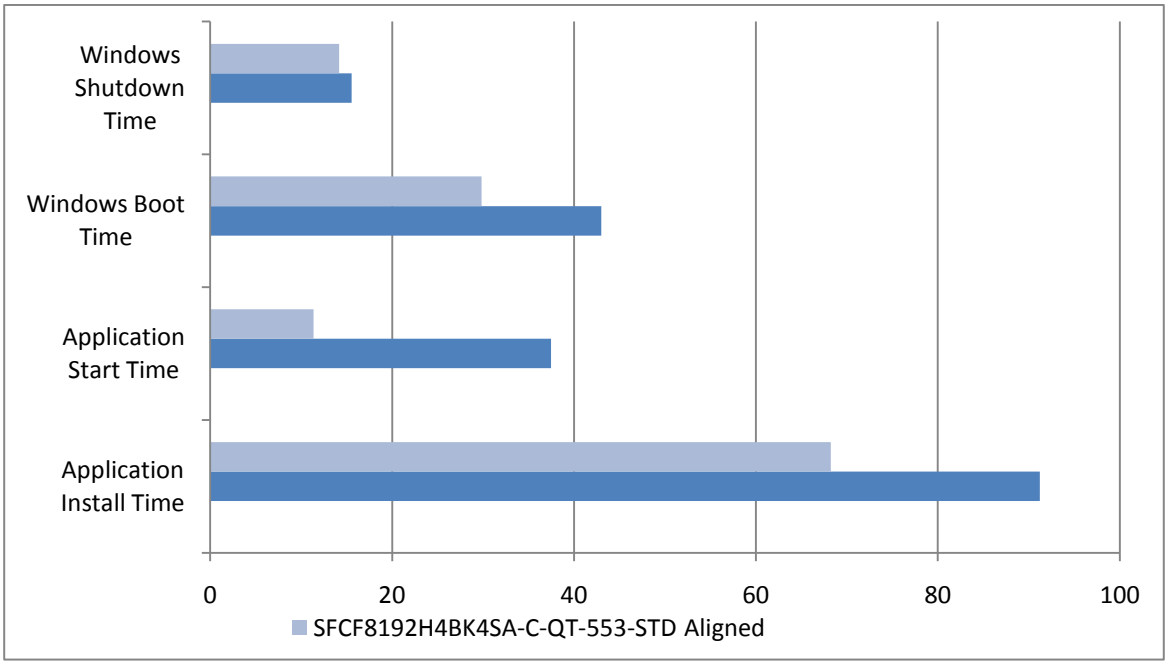
NAND Flash is organized in blocks and pages. All modern file systems also use blocks of sectors ("clusters") for organizing data storage. Data is primarily written in clusters.



If these clusters and logical pages are not aligned to each other, the device may have to erase and program more NAND blocks with each cluster write access. It thus is important to align the file system to NAND block boundaries. This will help reducing erase/program cycles (thus also reducing power fail risk) and will show better performance.

File system benchmarks showed that aligned file systems are up to 80% faster on heavy file system usage.

Graph: Performance comparison between aligned and unaligned FAT filesystems, measured using a Swissbit C-300 8GB CF card, PN: SFCF8192H4BK2SA*-553-*



For best life time, performance, and power fail stability, the customer should achieve the following low level goals:

- Write only the data really necessary
- Write optimized (big chunks, aligned to the devices Flash block / page boundaries)
- Make sure the hardware connection and power supply is optimal

3 Hardware design-in

3.1 Interface & Transfer Modus

For best performance and stability it is obviously important to make sure that the interface speed modus is optimal. Depending on the interface used, there are a number of points to be verified.

3.1.1 IDE / Parallel ATA

UDMA speed support

New devices supporting UDMA require a full UDMA capable main board, cable and connectors. Swissbit is able to supply devices with UDMA disabled for compatibility reasons.

For speed modes better than UDMA2, the ATA specification specifies the use of 80-conductor cables. These 80-conductor cables usually have one blue plug which must be used for the motherboard connector. The master device should be attached on the last connector of the cable (slave if present on the middle connector), to avoid signal reflections at the end of the cable.

Both the host chipset and the device will evaluate whether an 80-conductor cable is used. The device will report this in the ATA identify data (word 93, bit 13 (0x2000) must be 1 for 80-conductor cables).The operating system driver is obliged to reduce speed modes to UDMA2 if no such cable is found. If no cable is used (e.g. for onboard CompactFlash connectors), it may be necessary to override this driver behavior, depending on the design. Industrial BIOS often supply an option to override the host side detection.

Note that the CompactFlash specification requires a maximum cable length of 15cm and only one attached device (which must be configured as master).

Please consult the ATA specification, e.g. ATA/ATAPI-7 Volume 2, chapter "9.4 Host determination of cable type by detecting CBLID-", for more details and examples of a correct layout.

Data edge test

The IDE interface has 16 data lines, one word or 16 bits are transferred per cycle. Some systems have shown compatibility issues when all data lines switch their status at the same time.

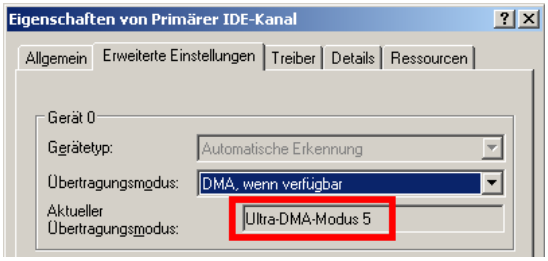
A data pattern file holding a repeated "0000 FFFF" data pattern is available for download[2] (unzipped 1MB). We recommend making copies of the file multiple times on the target system to validate that no transfer errors occur. After the test the device should still run in the same speed mode, and the content of the files should be validated.

Windows speed settings check

Under Windows the speed is controlled by the driver. The following instructions apply for the default Microsoft IDE driver only, for all Windows versions starting from Windows 2000.

The current speed mode of the device can be checked in the Windows device manger. Select "IDE ATA/ATAPI controller" -> "Primary IDE channel" or "Secondary IDE channel" -> ENTER -> Advanced Properties.

The speed is shown for the master (device 0) and slave (device 1) device.



When communication errors occur, Windows reduces the speed to ensure valid operation. It is difficult to manually restore the default speed again, but there is a small script available[3] that resets the speed mode recognition.

[2] ftp://public:public@office.swissbit.com/DataPatterns/0000FFFF-data.zip
[3] http://winhlp.com/tools/resetdma.vbs

The speed settings are also available in the registry.

Key: [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Class\{4D36E96A-E325-11CE-BFC1-08002BE10318}\00xx\]
Value Name: MasterDeviceTimingMode
Value Name: MasterDeviceTimingModeAllowed
Value Name: UserMasterDeviceTimingModeAllowed
Data Type: REG_DWORD

There are keys for each channel (00xx), and there are values for both master / device 0 (e.g. MasterDeviceTimingMode) or slave / device 1 (SlaveDeviceTimingMode).

The bits are used as follows:

PIO: Bits 0-7, 0x0000'0001 is PIO0, 0x0000'0002 is PIO1 etc.

MDMA: Bits 10-8, 0x0000'0100 is MDMA0, 0x0000'0200 is MDMA1, 0x00000400 is MDMA2

UDMA: Bits 17-11, 0x0000'0800 is UDMA0, 0x0000'1000 is UDMA1 etc.

Example: A MasterDeviceTimingMode of 0x0001'0010 means that PIO4 and UDMA5 are selected.

If Windows limits speed because of communication problems, it deactivates the bits in the appropriate *DeviceTimingModeAllowed value. The user can also deactivate speeds by adding a User*DeviceTimingModeAllowed value.

TRIM (Data Set Management)

For devices supporting the TRIM feature (e.g. CompactFlash C-4x0 models) set please see chapter 3.1.3 for details.

3.1.2 Serial ATA

Chipset mode (AHCI / Native Mode / Compatibility Mode)

Serial ATA can be operated in different modes. For backwards compatibility with old operating systems, the Serial ATA chipset often works in an emulation mode to simulate plain Parallel ATA (IDE). This mode is usually called "Compatibility mode" or "IDE mode" in the BIOS. While this mode often is simplest to use (no special drivers needed), it means that the operating system can't take advantage of SATA features like hot-plugging or NCQ (see below).

All SATA chipsets support a "native" or AHCI mode, which can be configured in the BIOS. This will need driver support by the operating system, but will often result in better performance. Microsoft Windows 7 provides out of the box AHCI drivers, earlier versions need vendor specific drivers to be installed. Please note that changing the SATA mode on an installed Microsoft Windows will often not work – it is recommended to choose AHCI mode before installation.

Devices supporting SATA II should be verified to run in SATA II speed if possible (depending on host support) – if a bad cable is used, the SATA speed could be downgraded. This can usually be checked with Swissbit Life Time Monitoring Tool (see chapter 7.1).

Native command queuing (NCQ) [4]

Native command queuing is an extension of the Serial ATA protocol, where multiple write or read commands are queued in the devices. This allows the storage device to internally optimize the order in which received read and write commands are executed, and to reduce latency by working on the SATA bus and Flash bus concurrently. The NCQ feature thus improves performance, especially for randomly distributed, relatively small read commands (e.g. when booting the operating system or loading an application).

Please note that to use NCQ, the SATA chipset must not run in compatibility mode (see above). All current desktop/server operating systems will automatically use NCQ if possible.

TRIM (Data Set Management)

For devices supporting the TRIM feature (e.g. CFast F-2xo models) set please see chapter 3.1.3 for details.

[4] http://en.wikipedia.org/wiki/Native_Command_Queuing

3.1.3 TRIM (Data Set Management)

TRIM commands allow an operating system to inform storage drives which blocks of data are no longer considered in use and can be wiped internally.

This information helps devices that support TRIM functionality to improve performance (especially for randomly distributed writes), and sometimes to reduce Flash stress (depending on the internal Flash management routines).

TRIM can be used on IDE and SATA links, AHCI is not a requirement.

For more, general information on TRIM please consult the extensive Wikipedia[5] article.

Operating system support

Trim is only effective on operating systems which support it. The table below identifies each notable operating system and the first version supporting the command.

Operating System	Supported since	Notes
Microsoft Windows	NT 6.1 (Windows 7 and Windows Server 2008 R2) – October 2009	(see notes below)
Linux-kernel	2.6.33	Support for the ATA TRIM command was added in 2.6.33. Not all filesystems make use of TRIM. Among the filesystems that can issue TRIM requests automatically are Ext4, Btrfs, FAT, GFS2 and XFS. However, this is disabled by default due to performance concerns, but can be enabled by setting the "discard" mount option. Ext3, NILFS2 and OCFS2 offer ioctls to perform offline trimming.
Mac OS X	10.6.8 – 23 June 2011	Although the AHCI block device driver gained the ability to display whether a device supports the TRIM operation in 10.6.6 (10J3210), the functionality itself remained inaccessible until 10.6.8, when the TRIM operation was exposed via the IOStorageFamily and filesystem (HFS+) support was added. Only enabled for Apple-supplied SSDs. TRIM can be enabled for other devices by modifying a kernel extension.
DragonFly BSD	May 2011	
FreeBSD	8.1 – July 2010	Filesystem support was added in FreeBSD 8.3 and FreeBSD 9, beginning with UFS.
OpenSolaris	July 2010	

Windows specialities

Requirements for Windows TRIM support:

- Windows 7 or newer
- Host Controller in IDE (Compatibility) or AHCI mode (or RAID, but only with Intel chipset & Intel drivers)
- TRIM capable driver:
  - Microsoft drivers (IDE or AHCI)
  - Intel RST (Rapid Storage Technology) since 9.6.0.1014
  - AMD AHCI drivers since 1.2.1.263

TRIM is supported for both IDE (CompactFlash) or SATA (SSD, CFast) devices. AHCI is not a requirement. It is supported for all Microsoft file systems (FAT, NTFS, exFAT).

[5] http://en.wikipedia.org/wiki/TRIM

SATA Bridges

At the time of writing, there are no known SATA bridges (SATA to IDE or IDE to SATA) that support the TRIM command. Issuing a TRIM command will lead to a failure / timeout on the host side (operating system).

To work around this, the operating system should be configured to not use TRIM.

Disabling TRIM on Windows 7

Open a command line with administrator rights (Start – type cmd – Press Ctrl+Shift-Enter)

Disable TRIM:

```
fsutil behavior set DisableDeleteNotify 1
```

Enable TRIM (default):

```
fsutil behavior set DisableDeleteNotify 0
```

Display current setting:

```
fsutil behavior query DisableDeleteNotify
```

That it's enabled (= not disabled) does not mean that is actually is used (unlike stated on some internet sources).

Verifying in-system TRIM functionality

Verifying that the TRIM feature is really used by a system is not a trivial task, especially for Microsoft Windows systems.

Swissbit devices that support TRIM functionality include a SMART feature to check if the TRIM functionality is used by the operating system. This provides a measure (percentage) of how much of the drive is currently marked as free (trimmed). Once the card is filled with data, this TRIM usage percentage should decrease (going near 1%). If the user data is then deleted, the TRIM usage percentage should rise again (> 90%).

As other SMART values this can be read using the Swissbit Life Time Monitoring tool, see chapter 7.1 for details.

3.1.4 SD / µSD / MMC

SD (µSD) and MMC cards can operate in SD/MMC mode or SPI interface.

Swissbit SD (µSD cards) can work with 1 or 4 bidirectional data lines with up to 50MHz clock frequency. Swissbit MMC cards work with 1 bidirectional data line with up to 52MHz clock frequency. Both card types could also work in SPI mode with 1 input and 1 output data line with up to 50/52MHz.

In SPI mode the host can handle multiple cards at the same bus, that will be distinguished with a CE line for each card.

3.2 Power supply considerations

All Swissbit Flash storage devices use a built-in voltage detector to detect power failure. The controller is immediately reset and the Flash memory components are write-protected. When a write command is interrupted, the data modified by this command may be lost. The exact firmware power fail protection algorithms of the devices may depend on the product group / firmware revision.

For Flash storage devices without a write cache, the data is correctly written in the Flash memory components as soon as the write command is confirmed. Storage devices with a write cache may require an additional cache flush command. If the operating system can detect imminent power failure, at least the current write command should be completed up to the point where the device confirms the command. If the time frame to power failure is big enough, all caches used should be synchronized to the device.

Swissbit advises testing power fail stability of the overall system with test installation or simulation where power failure is simulated repeatedly.

3.3 Temperature

Most Swissbit Flash product lines are offered in 3 different temperature grades:

- Commercial grade 0...+70°C
- Extended grade -25...+85°C
- Industrial grade -40...+85°C

The extended and industrial grade memories use special components and are extensively tested at low and high temperature in production (100% screening).

Temperature can influence the driver strengths of the interface drivers, so wave forms, timings, and current can vary with temperature and should be verified in the target system.

Devices shall not be operated outside the specified temperature range.

4 Optimal partitioning

In chapter 2 we showed that it is important to align the file systems blocks (clusters) to the logical page boundaries of the device. The logical pages are currently max 64kB / 128 sectors of size, but we recommend aligning to 256kB / 512 sectors for future safety.

All modern computer systems allow dividing a disk device into multiple partitions. It thus is the first step to make sure these partitions are aligned correctly. For most file systems it is optimal to align to multiples of 64kB (or 128kB, 256kB, etc). When using smaller file system cluster sizes, it will suffice to align at cluster size boundaries.

For other file systems, especially FAT, the partitions should be aligned with an offset, see chapter "5 – Choosing and optimizing the file system" for more details.

Swissbit Flash storage devices are delivered with an IBM PC-compatible partition table (Master Boot Record, MBR), with one big partition covering the whole disk. In most cases these standard partitions are not well aligned, for best compatibility with old systems.

SD / microSD / MMC card specific

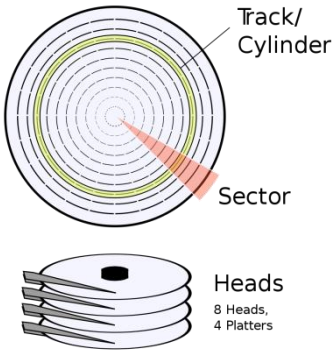
SD, microSD & MMC cards are always delivered with aligned FAT partitions, according to the specifications. Please also see "4.3 SD/MMC card specific" below for more information.

4.1 CHS vs. LBA addressing

Cylinder-head-sector (CHS) addressing was an early method for giving addresses to each physical block of data on a hard disk drive. Of course this is not applicable to NAND Flash based devices, so the controller will supply pseudo CHS values which are then translated to Logical Block Addresses (LBA).

Though CHS values no longer have a direct physical relationship to the data stored on disks, pseudo CHS values are still being used by many utility programs and operating systems.

The 16-byte entries within an MBR Partition Table have CHS-tuples which are limited to a total of 1024 cylinders, 255 heads and 63 sectors (about 7.8 GiB).



Logical block addressing (LBA) in contrast is a common scheme used for specifying the location of blocks of data stored on storage devices. LBA is a particularly simple linear addressing scheme; blocks are located by an index, with the first block being LBA=0, the second LBA=1, and so on.

The "block" term is used for the smallest addressable data size in this context, it is not related to NAND Flash blocks. These logical blocks currently always are of size 512 Bytes (one sector) for Swissbit Flash devices.

4.2 Partitioning tools examples

Warning: please use these tools with care. Any existing data on the disks worked on will be lost!

4.2.1 Windows

Windows versions starting from Vista (including 2008 Server and Windows 7) will by default align partitions 2048 sectors / 1MB. When using one of these versions, the default Windows disk management tools can be used (when using NTFS).

Earlier Windows versions (e.g. 2000, XP, 2003 Server) do not align partitions by default. Microsoft supplies partitioning tools to allow more flexible partitioning.

To check existing partitions use the system information tool (Start – Run – msinfo32). In MSInfo: Components -> Storage -> Disks -> Partition Starting Offset. This is the offset in Bytes, divide by 512 to get the start sector.

Recommended for older Windows versions (2000, XP): DiskPar tool

Microsoft supplies a disk partitioning tool for older Windows versions, "diskpar" (please note the intentionally missing "t"). The diskpar tool is available from the Microsoft Homepage as part of the Windows 2000 Resource Kit, or directly from Swissbit[6].

Use this tool with the following steps at a command prompt:

- Use diskpar –i X to show drive information, where X is the Windows drive number (see DiskPart above for a way to get the drive number)
- Use diskpar –s X to create a new partition. The tool will query parameters on the command prompt. It requires the offset to be specified in number of sectors.

```
C:\>diskpar -s 2
Set partition can only be done on a raw drive.
You can use Disk Manager to delete all existing partitions
Are you sure drive 2 is a raw device without any partition? (Y/N) y

---- Drive 2 Geometry Infomation ----
Cylinders = 124
TracksPerCylinder = 255
SectorsPerTrack = 63
BytesPerSector = 512
DiskSize = 1019934720 (Bytes) = 972 (MB)

We are going to set the new disk partition.
All data on this drive will be lost. continue (Y/N)? y
Please specify starting offset (in sectors): 2048
Please specify partition length (in MB) (Max = 971):
...
```

[6] ftp://public:public@office.swissbit.com/SFxx_Tools/diskpar.exe

Recommended for Windows 2003 and newer: DiskPart

The DiskPart tool is part of the Windows distribution. It can only be used with fixed disks and does not support removable devices.

Only newer versions of DiskPart allow aligning partitions. The new version is supplied starting from Windows 2003 SP1, older versions (as for example delivered with Windows XP) don't support manual aligning of partitions. For Windows 2003 without ServicePack the tool is available standalone as a hotfix download [7].

Use this tool with the following steps:

- At a command prompt, type diskpart, and then press ENTER. The diskpart prompt will appear (DISKPART>). Type "list disk" then press ENTER. All fixed disks in the system will be listed.

```
DISKPART> list disk

  Disk      Status      Size    Free
  --------  ----------  -------  -------
       0    Online        75 GB      0 B
       1    Online        76 GB      0 B
       2    Online       973 MB      0 B
```

- Select the disk you want to work on with "select disk X" where X is the disk number.
- Optional: to see the currently defined partitions use "list partition". Use "clean" to delete all defined partitions.
- Create a new partition with the command "create partition primary align=64" with alignment to 64kB (128 sectors).

4.2.2 Linux

Under Linux there are different tools available to view and create partitions, e.g. fdisk, cfdisk, sfdisk, parted, ... Each tool has a different feature set and its pro's and con's.

In this document we will use the fdisk tool, which is part of the "util-linux-ng" package.

Start fdisk as root with the disk you want to work on as a parameter:

```
# fdisk -lu /dev/sdX
Disk /dev/sdX: 1024 MB, 1024966656 bytes
16 heads, 63 sectors/track, 1986 cylinders, total 2001888 sectors
Units = sectors of 1 * 512 = 512 bytes
Disk identifier: 0x7d04cbb0

   Device Boot      Start         End      Blocks   Id  System
/dev/sdX1              63     2001887     1000912+  83  Linux
```

This partition is starting on sector 63, which is misaligned.

[7] http://support.microsoft.com/kb/923076/

It is possible to create aligned partitions by specifying the values in LBA sectors. For better interoperability with other operating systems, it may be necessary to override the tools automatically detected physical CHS values with Pseudo-CHS values that will allow round alignment with CHS addressing, e.g. specify the parameters –S 32 –H 16.

Newer versions of partitioning tools under Linux will automatically align partitions, this workaround is not needed.

```
# fdisk -u -S 32 -H 16 /dev/sdX
[...]

Command (m for help): n
Command action
   e   extended
   p   primary partition (1-4)
p
Partition number (1-4): 1
First sector (32-4001759, default 32): 256
Last sector, +sectors or +size{K,M,G} (256-4001759, default 4001759):
Using default value 4001759

Command (m for help): w
The partition table has been altered!

Calling ioctl() to re-read partition table.
Syncing disks.


# fdisk -lu /dev/sdX

Disk /dev/sdX: 2048 MB, 2048901120 bytes
16 heads, 32 sectors/track, 7815 cylinders, total 4001760 sectors
Units = sectors of 1 * 512 = 512 bytes
Disk identifier: 0xf32a44f9

   Device Boot      Start         End      Blocks   Id  System
/dev/sda1             256     4001759     2000752   83  Linux


# fdisk -l /dev/sdX

Disk /dev/sdX: 2048 MB, 2048901120 bytes
16 heads, 32 sectors/track, 7815 cylinders
Units = cylinders of 512 * 512 = 262144 bytes
Disk identifier: 0xf32a44f9

   Device Boot      Start         End      Blocks   Id  System
/dev/sdX1               1        7816     2000752   83  Linux
```

To start over with a clean partition table, overwrite the MBR sector with all zero:

```
# dd if=/dev/zero of=/dev/sdX bs=1b count=1
```

To check alignment of a FAT clusters under Linux, we recommend using the fsstat utility, which is usually distributed in the "sleuthkit" package.

Example:

```
# fsstat /dev/sdX1

FILE SYSTEM INFORMATION
--------------------------------------------
File System Type: FAT32

OEM Name: mkdosfs
Volume ID: 0x4705a8a6
Volume Label (Boot Sector): SWISSMEMORY
Volume Label (Root Directory): SWISSMEMORY
File System Type Label: FAT32
Next Free Sector (FS Info): 7632
Free Sector Count (FS Info): 3889704

Sectors before file system: 0

File System Layout (in sectors)
Total Range: 0 - 3897343
* Reserved: 0 - 31
** Boot Sector: 0
** FS Info Sector: 1
** Backup Boot Sector: 6
* FAT 0: 32 - 3831
* FAT 1: 3832 - 7631
* Data Area: 7632 - 3897343
** Cluster Area: 7632 - 3897343
*** Root Directory: 7632 - 7639

METADATA INFORMATION
--------------------------------------------
Range: 2 - 62235398
Root Directory: 2

CONTENT INFORMATION
--------------------------------------------
Sector Size: 512
Cluster Size: 4096
Total Cluster Range: 2 - 486215

FAT CONTENTS (in sectors)
--------------------------------------------
7632-7639 (8) -> EOF
```

Please note that the LBA numbers given by the tool are relative to the partition offset.

To find the alignment of the FAT clusters use the following calculation:

First Cluster LBA = Partition Offset + Cluster Area Start

For the example above (1024 would be the partition offset seen by fdisk):

First Cluster LBA = 1024 + 7632 = 8'656

The cluster size is 4'096 Bytes, sector size 512→ 4096 / 512 = 8 sectors.

8'656 is dividable by 8 sectors without rest → the file system is aligned.

4.3 SD/MMC card specific

The SD/MMC standards define the partitioning and formatting of the devices, in a way which guarantees that the file system clusters are aligned to the logical page boundaries.

The SD Association provides a free partitioning/formatting tool for Windows ("SD Formatter"). Please consult the SD Association homepage[8] for more details. The current link to the download page is https://www.sdcard.org/consumers/formatter_3/. If this page is not available please search for "SD formatter".

Please do not use standard Microsoft Windows tools for these product groups, as this will almost always result in misaligned clusters.

[8] http://www.sdcard.org

5 Choosing and optimizing the file system

The file system chosen for a system can have a big impact on power fail stability, performance, and life time of a device. In this chapter we will explain basic differences and pro & contras of often used file systems.

There are many different file systems available, depending on the operating system used. We will therefore discuss the options individually for the most often used operating systems.

5.1 Basic file system properties

5.1.1 Journaling vs. non-journaling file systems

A journaling file system is a file system that keeps track of the changes it intends to make in a journal (usually a circular log in a dedicated area of the file system) before committing them to the main file system. In the event of a system crash or power failure, such file systems are faster to bring back online and less likely to become corrupted.

Basic or older file systems (for example FAT) do not use journals and thus are much more prone to power fail errors leading to corrupt file system structures.

While power fail stability is greatly improved by journaling file systems, the additional journal writing means more write access to the device – thus has the disadvantage of reduced performance and life time drawbacks. Still we recommend using journaling file systems if the target system is not protected against power failure and relies on a stable file system.

5.1.2 Cluster / block sizes

All file systems work with groups of sectors, called clusters or blocks. Typical cluster sizes range from 1 sector (512 Bytes) to 128 sectors (64kB), most often 2kB, 4kB or 8kB depending on device capacity.

When using a cluster size of 4kB, a file with size 4.5kB is stored in 2 clusters and actually takes 8kB of disk space. These two clusters do not necessarily have to be neighbors, one cluster may be located in the beginning of the partition, the other one at the end.

For performance reasons it is generally good to choose the cluster size bigger than default. The disadvantage of big clusters is the disk space that is lost if there are lots of small files in the file system ("internal fragmentation" or "slack space").

When formatting with Windows, we recommend using the Disk Management for the task – it is more flexible for choosing cluster sizes.

5.1.3 Quick format vs. full format

Often formatting tools allow choosing between "full format" and "quick format". The only difference between the two options is that full format first checks the whole device for defect sectors. Defect sectors can occur with traditional hard disks, but not with Flash devices – as defect NAND Flash blocks are automatically and transparently replaced by spare blocks.

As quick format is much faster and does not have any drawbacks for Flash devices, it is safe and recommended to use quick format even in production environments.

5.2 Windows (Desktop / Server / CE)

Desktop/Server Windows versions (from NT up to Windows 7, including the embedded variants) traditionally only support two Microsoft file systems: FAT (non-journaling) and NTFS (journaling). Newer Windows versions also support exFAT (non-journaling) which is optimized for Flash devices.

This means that with Windows, the main decision is whether to use a journaling or a non-journaling file system (see chapter 5.1.1 more details). As NTFS also has many other advantages over FAT, it often is the recommended choice.

5.2.1 NTFS

NTFS (New Technology File System) is the default file system for Windows desktop and server systems.

NTFS has many improvements over FAT such as improved support for metadata and the use of advanced data structures to improve performance, reliability, and disk space utilization, plus additional extensions such as security access control lists (ACL) and file system journaling.

As NTFS is a journaling file system, it is very stable in regards to power fail.

NTFS Partition / Cluster Alignment: the partition should be aligned to page boundaries. The file system clusters are automatically aligned correctly.

5.2.2 FAT

FAT (File Allocation Table) is probably the most widely used and supported file system and was designed by Microsoft. It is quite simplistic in its feature set and does not support journaling. As we explained above it thus is more prone to power fail errors damaging file system structures.

It is very important to always check and repair FAT file systems on boot. Desktop Windows variants do this automatically; the embedded variants (e.g. Windows XP Embedded) do not. Please see "6.1.3 File system repair on boot" for more details.

FAT Partition / Cluster Alignment: the first management sectors of FAT (Partition Boot Record, Reserved Sectors, FAT1, FAT2, Root Directory) may take any amount of sectors. Even if the partition is aligned, it is not guaranteed that the clusters are aligned.

There are different ways on achieving this:

- Adjusting the number of "reserved sectors" in the FAT structures.

  For FAT16 by default this is only 1 sector (PBR), adjusting this value is not recommended by Microsoft for compatibility reasons – still it may be good enough for enclosed systems, as Windows can work with any number of reserved sectors.

  For FAT32 the number of reserved sectors by default is 32, and can be adjusted to higher values without introducing compatibility issues. One tool allowing this is the Linux mkdosfs (parameter -R).

  There also are specialized formatting tools which will always align clusters to 4kB boundaries. One of these tools is "oformat", delivered by Microsoft with the Windows deployment toolset. Another example is the FreeDOS tool "format" (specify parameter /a). These are a good alternative if the cluster size was chosen as 2kB or 4kB, but not for bigger cluster sizes.

- Adjusting the partition offset so that the clusters are aligned. This is more complex, but will work with any formatting tool and any file system.

  For this approach, create a normal partition and format with the tools of your choice. Then use a tool to find the address of the first data cluster. We recommend the Windows tool WinHex for this, which is available as a free trial version[9].

  1. Open the disk (Tools – Open disk, Physical Media). WinHex will show the partitions created including the 1st sector:

     Hard disk 2

     Partitioning style: MBR

Name	Ext.	Size	Created	Modified	Accessed	Attr.	1st sector
Partition 1	FAT16	3.8 GB					63
Start sectors		31.5 KB					0
Unpartitionable space		5.5 MB					8016435

     In this example there is one partition starting at sector 63 (misaligned).

  2. Double-click partition 1. The tool will show the first data cluster in the left grey area:

     First data sector: 528

     Please note that this value is relative to the partition start, so the first data cluster is located at sector 528 + 63 = 591.

     The tool also shows the cluster size:

     Bytes per cluster: 65'536

     65536 bytes (64kB) divided by 512 (sector size) is 128 sectors.

     This means that the start address of 591 should be moved to a multiple of 128 (the smaller of cluster size 128 sectors and alignment size of recommended 512 sectors). For this example we choose 640. Therefore the partition should start at sector 640 – 528 = 112.

  3. We use the diskpar tool as shown in chapter 4.2.1:

```
C:\> diskpar -s 2
Set partition can only be done on a raw drive.
You can use Disk Manager to delete all existing partitions
Are you sure drive 2 is a raw device without any partition? (Y/N) Y

---- Drive 2 Geometry Infomation ----
Cylinders = 499
TracksPerCylinder = 255
SectorsPerTrack = 63
BytesPerSector = 512
DiskSize = 4104414720 (Bytes) = 3914 (MB)

We are going to set the new disk partition.
All data on this drive will be lost. continue (Y/N)? Y

Please specify starting offset (in sectors): 112
Please specify partition length (in MB) (Max = 3914): 3914

Done setting partition.
---- New Partition information ----
StatringOffset = 57344
PartitionLength = 4104126464
HiddenSectors = 112
PartitionNumber = 1
PartitionType = 7

You now should use Disk Manager to format this partition
```

  4. Format the new partition again. Use steps 1 to 3 to verify correct cluster alignment.

[9] http://www.winhex.com

5.2.3 exFAT

exFAT is an incompatible replacement for FAT file systems that was introduced with Windows Embedded CE 6.0. It is explicitly optimized for Flash memory components.

Microsoft has offered a hotfix to add support for exFAT to Windows XP, while Windows Vista Service Pack 1 added exFAT support to Windows Vista. exFAT offers a few improvements over FAT file systems, but still does not support journaling. We recommend exFAT over FAT for Flash devices, but compatibility with other systems (customer support, ...) may be an obstacle.

exFAT Partition / Cluster Alignment: the partition should be aligned to page boundaries. The file system clusters are automatically aligned correctly.

5.2.4 TFAT / TexFAT

TFAT and TexFAT are layers over the FAT and exFAT file systems respectively that provide a level of transaction safety to reduce the risk of data loss in the event of a power outage or unexpected removal of the drive. These may be good options if your target system supports these file systems.

5.3 Linux

There are many different file systems available for Linux. It would be out of scope to describe and compare the details of all file systems in detail – each file system has its pros and contras. In general we recommend choosing a well established and tested file system like the ext* family, XFS or JFS.

5.3.1 Non-journaling file systems

ext2 is the most common non-journaling file system. It is very well tested and can be recommended for applications not requiring extended power fail stability.

ext2 Partition / Cluster Alignment: the partition should be aligned to page boundaries. The file system clusters are automatically aligned correctly.

5.3.2 Journaling file systems

ext3 and ext4 are journaling file systems that are commonly used with the Linux kernel. These are the default file systems for many popular Linux distributions. Especially ext3 is very well tested and can be recommended as a safe choice. The successor ext4 adds many modern file system features and is continuously more often used.

ext3/ext4 support different levels of journaling: Journal (lowest risk), Ordered (medium risk, default), and Writeback (high risk). The journal mode to be used is specified as a mount option. We recommend using Ordered or Journal. The default depends on distribution / kernel. Check the kernel log (dmesg) for the journal currently used.

```
EXT3-fs (sdb1): using internal journal
EXT3-fs (sdb1): recovery complete
EXT3-fs (sdb1): mounted filesystem with writeback data mode
```

ext3/ext4 Partition / Cluster Alignment: the partition should be aligned to page boundaries. The file system clusters are automatically aligned correctly.

XFS and JFS are journaling file system created by Silicon Graphics respectively IBM. Both are known to be very stable in power fail situations.

XFS Partition / Cluster Alignment: the partition should be aligned to page boundaries. The file system clusters are automatically aligned correctly.

5.3.3 Log-structured file systems

Log-structured filesystem are file systems designed for high write throughput. All updates to data and metadata are written sequentially to a continuous stream, called a log. In theory this approach is optimal for NAND Flash based devices.

An example for a log-structured file system is **NILFS2**, which was introduced in Linux Kernel 2.6.30.

5.3.4 Special Raw Flash file systems

There are some Linux file systems that are designed to be used with NAND Flash devices, for example JFFS, LogFS, UBIFS or YAFFS. It is very important to know that these file systems are meant to be used directly on NAND Flash components – they implement functions like wear leveling which are also implemented by the Flash controller.

When a Flash device has a built-in controller that implements wear leveling, it is contra-productive to use one of the file systems above. We recommend using one of the standard file systems instead.

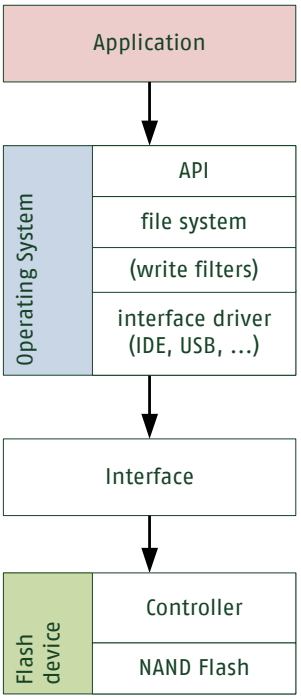
6 Operating system configuration

There are a lot of layers between the application and the actual Flash device. The operating system supplies an abstract application programming interface (API) for the application, implements a file system, and an interface driver.

The number, size and location of write accesses on the device thus depend on a lot of different influences – not only the application, but also file system and interface driver (mainly caches).

Not only the application but also the operating system (including file system) has a big influence on the Flash usage. In this chapter we will discuss the most useful options that widely used operating systems supply to optimize lifetime, performance and power fail stability with Flash devices.

Often the operating system also checks whether the device is "fixed" or "removable", and then uses different write caching parameters.



6.1 Windows

6.1.1 File timestamps tweaking

File systems track the date and time a file was created, last written, and last read (accessed). In almost every system the last access time is not necessary for operation, but will be written on the device even if the files are only read.

The updating of the last access time can be disabled for NTFS with a registry key:

Key: [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\FileSystem]
Value Name: NtfsDisableLastAccessUpdate
Data Type: REG_DWORD
Value Data: 0 = disable, 1 = enable

Or use the NTFS customizing tool "fsutil" on the command line:

```
fsutil behavior set disablelastaccess 1
```

Query the current mode with fsutil:

```
fsutil behavior query disablelastaccess
```

Microsoft changed the behavior starting from Windows 2008 Server and now disables the last access update by default.

As FAT only store the last access date (no time), this feature will only have a very minor effect on write cycles. Microsoft does not support disabling the feature for FAT.

exFAT supports the full last access time, but currently there is no way known to configure the behavior for this file system.

6.1.2 Write filters (Windows Embedded only)

Windows Embedded supports two different write filters that provide the ability to write-protect a whole partition (Enhanced Write Filter, EWF), or only some files (File Based Write Filter, FBWF).

Instead of writing the data to the card, the data written is transparently kept in RAM and will be lost on power cycles or reboot. This can be very useful especially for protecting the Windows system partition from modifications or power failure issues. Only the data really necessary (e.g. log files, configuration files) are actually written on the Flash device.

Swissbit strongly recommends using write filters to reduce write operations.

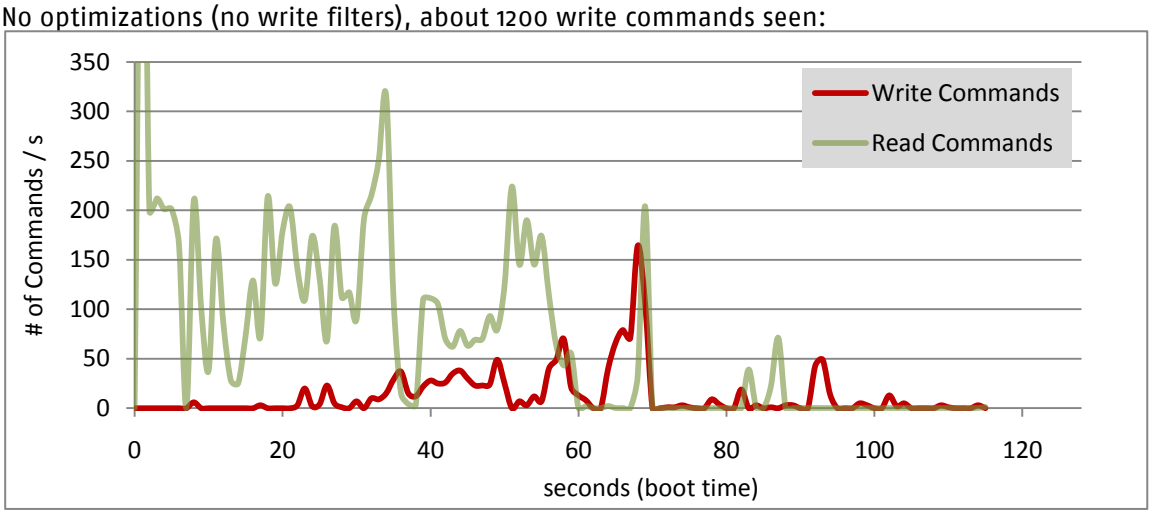
Please see the Microsoft documentation (EWF[10], FBWF[11]) for more details on filter usage.

[10] http://msdn.microsoft.com/en-us/library/ms912906%28WinEmbedded.5%29.aspx

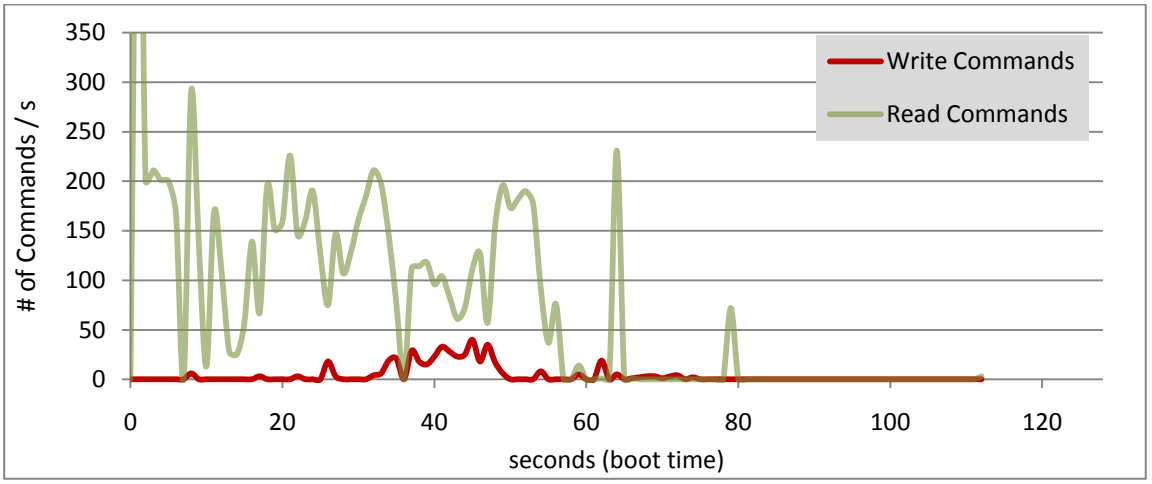
[11] http://msdn.microsoft.com/en-us/library/aa940926%28WinEmbedded.5%29.aspx

The following charts illustrate the number of write and read commands per second on a typical Windows XP Embedded boot from an NTFS partition (including a short file copy action on writeable files).

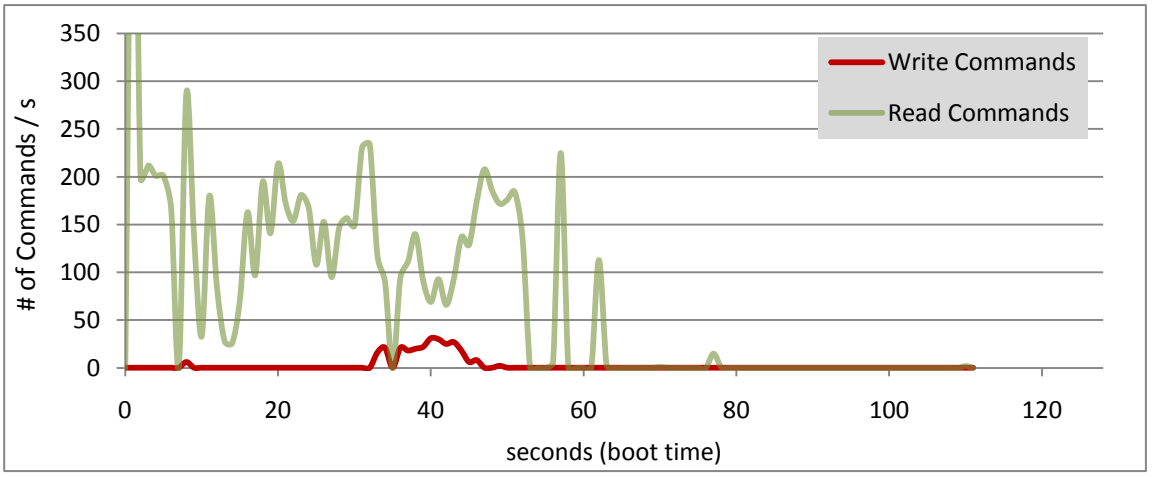
No optimizations (no write filters), about 1200 write commands seen:



File based write filter active, about 400-500 write command seen:



File based write filter active and NTFS last access update disabled, about 250-300 write commands:



6.1.3 File system repair on boot

Windows dektop / server systems execute a file system check on boot, if a file system is marked as "dirty" after power fail. This is important for file system stability after power fail, and strongly recommended by Swissbit.

Windows Embedded does not do this by default, probably because of the Enhanced Write Filter (see previous chapter). If the EWF filter is activated on a partition, this partition should be excluded from the file system check, as the check will only use boot time and RAM (file system cache) with no advantages.

The behavior is controlled by a registry key "BootExecute", and should hold an "autocheck ..." command line a follows:

Key: [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager]
Value Name: BootExecute
Data Type: REG_MULTI_SZ
Value Data: autocheck autochk *

In this variation all file systems are checked at boot time.

The registry key can be created by the chkntfs tool (options work regardless of the file system FAT / NTFS), use /D for the default behavior of checking all file systems:

```
chkntfs /D
```

To exclude a EWF protected file system from the check use the parameter /X:

```
chkntfs /X F:
```

This will change the registry key above to exclude F:\ from the automatic check: "autocheck autochk /k:F *".

6.1.4 Disable unnecessary services / background software

Please look at all activated software services or automatically started software and let only necessary services run automatically. One example is the Windows Search service (or Indexing service), which is often not necessary. The Autoruns tool[12] by Microsoft can help with this.

The Disk Defragmenter service (Windows 7) should be disabled if only Flash storage is used in a system, as defragmenting does not improve performance for Flash based devices and as such leads to much unnecessary data written.

Windows 7 will turn off defragmenting automatically if the device is fast enough.

6.1.5 Disable drive indexing

Drive indexing should be turned off if not explicitly needed for regular file searches.

Click on my computer, right click on the Flash device, Properties, un-check "Drive indexing", a wizard will appear, follow through with the Wizard and let it finish.

To disable drive indexing globally (for all drives) the Indexing service can be disabled, making the steps above not necessary.

6.1.6 Disable Prefetching

Prefetching is a technology that tries to optimize file loading to gain better performance for Windows and application starts. The performance gain for Flash devices is expected to be minor, so it should be considered to turn prefetching off to save write commands.

The basic service used starting from Windows XP is called Prefetch and can be turned off only by modifying a registry value[13].

Key: [HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Control\Session Manager\Memory Management\PrefetchParameters]

[12] http://technet.microsoft.com/en-us/sysinternals/bb963902.aspx
[13] http://msdn.microsoft.com/en-us/library/ms940847%28WinEmbedded.5%29.aspx

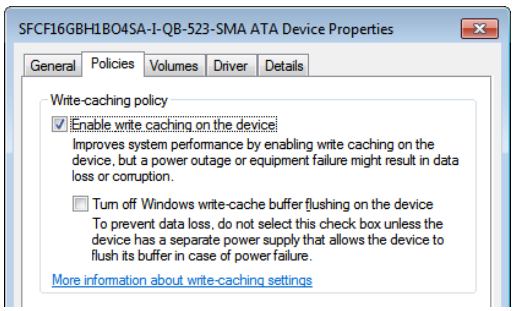
Value Name: EnablePrefetcher
Data Type: REG_DWORD
Value Data: 0

Windows Vista / 7 add a technology called Superfetch on top of Prefetch. As the superfetch only introduces very few write commands, it can usually be left turned on. To disable Superfetch manually, disable the Superfetch Windows service (and disable prefetch as shown before).

Windows 7 will automatically turn off Superfetch, if the device is considered fast enough.

6.1.7 Disk write cache policy

The disk write cache can be configured per device in the device manager (select drive -> Properties -> Policies). Swissbit generally recommends enabling the write cache.



Disabling the write cache will have a major impact on the drive performance and will increase the number of writes to the device (negative impact on Flash lifetime). The advantage of disabling the write cache is mainly in reduced data loss of the software applications running on the system.

Sensitive software (e.g. database services) usually requires a "data flush" from the operating system after writing sensitive data. This data will then be flushed from the cache.

Enabling the write cache usually has no disadvantages for the stability of the file system (regarding file system corruptions).

Turning off Windows "write-cache buffer flushing" (second checkbox) will improve performance, but should only be used if the system is protected against power failure.

6.1.8 IDE master / slave configurations

If only one device is attached on an IDE bus, the device must be configured as master. Windows will produce a bluescreen error when waking up from standby/hibernate when no master is present – this is a known issue[14] of Windows.

6.2 Windows CE

6.2.1 File and disk caching

Windows CE supports a number of features that allow file and disk based caching. It is recommended to use caches to improve performance and reduce the amount of data written to the device.

Please see Microsoft documentation[15] for more details. There are registry settings available for controlling the cache sizes, see Microsoft Documentation[16].

We recommend testing the power fail stability of the overall system when using extensive caching.

[14] http://support.microsoft.com/kb/330100/
[15] http://msdn.microsoft.com/en-us/library/aa910672.aspx
[16] http://msdn.microsoft.com/en-us/library/aa914334.aspx

6.2.2 RAM based file system

Windows CE supports a RAM based file system, which should be used for all temporary data that is not needed after a power failure. This can very efficiently reduce the amount data written on the device.

Please see Microsoft documentation[17] for more details.

[17] http://msdn.microsoft.com/en-us/library/aa915502.aspx

6.3 Linux

6.3.1 File system options

For Linux there are a number of options available when mounting file systems. These can be given manually for the mount command (parameter -o), or be defined in the /etc/fstab file. The manual page for mount lists the available options for most file systems.

General recommended options are:

- async (default): asynchronous mode will limit the number of writings. Sync mode (disables caches) is not recommended for Flash devices!
- noatime: do not update file "last access" times. This will reduce write operations massively on heavy file read usage.
- nodiratime: do not update directory "last access" times. As noatime this will further reduce write operations.

Recommended for ext3:

- data=?: you may want to use an alternative journaling mode for file data. See "5.3.2 Journaling file systems" and ext3 documentation for details.
  To use modes other than ordered on the root file system, pass the mode to the kernel as boot parameter, e.g. rootflags=data=ordered

Recommended for XFS:

- nobarrier: XFS by default uses write barriers to make sure the controller (e.g. RAID controller) and disk cache are flushed where necessary, to ensure file system consistency on power failure. As current Swissbit Flash storage devices do not use a RAM cache (data is always written on Flash immediately), and when no special cached ATA controller is used, the barrier can safely be turned off. This will improve performance as unnecessary cache sync commands are not issued.
  We recommend extensive power fail testing to make sure the feature is not needed. See the XFS documentation[18] for more details.

6.3.2 I/O scheduler

Linux supports different disk I/O schedulers. The default that is used depends on the distribution and/or kernel version.

According to reports the simple "NOOP" scheduler provides best performance on Flash devices. This has not been verified by Swissbit, but may be worth consideration.

The default scheduler can be configured when building the kernel (in the configuration menu at location Enable the block layer -> IO Schedulers -> Default I/O scheduler).

The scheduler can also be chosen at runtime for each disk device:

```
echo noop > /sys/block/hda/queue/scheduler
```

Please consult the appropriate documentation for more details.

6.3.3 RAM based file system

Standard Linux distributions do not support write filters, but allows use of an In-RAM file system. These file systems can be used for temporary files that can be lost on power failure.

The most often used file system used for this is tmpfs. Swissbit recommends using this wherever possible.

For more details on tmpfs please consult the Linux documentation[19].

[18] http://xfs.org/index.php/XFS_FAQ
[19] http://lxr.linux.no/#linux+v2.6.34/Documentation/filesystems/tmpfs.txt

7 Life time monitoring & assessment

All Swissbit Flash storage products allow reading out Flash status information. The level of detail and the access method depend on the product and model used.

Please consult the data sheets for more information.

7.1 Swissbit Life Time Monitoring for Windows & Linux

7.1.1 Overview

Swissbit provides a free, fully featured life time monitoring tool for Windows and for Linux to read out device status information for most products.

The tool supports the following product groups:

- CompactFlash cards C-300 or newer
- CFast cards (all product groups)
- SSD, mSATA, slimSATA (all product groups)
- USB devices (all product groups)

Other S.M.A.R.T. software may be used, but will not interpret all Swissbit custom values. Please consult the datasheet and the SBLTM tool documentation for a description of all values available.

The capabilities of each product group and product generation can vary.

7.1.2 Download

Windows:

ftp://public:public@office.swissbit.com/SFxx_LTM_Tool/SBLTM-Windows-32bit-LATEST.zip

Linux x86-32 (32 bit):

ftp://public:public@office.swissbit.com/SFxx_LTM_Tool/SBLTM-Linux-x86-32-LATEST.tar.gz

Linux x86-64 (64 bit, architecture "amd64"):

ftp://public:public@office.swissbit.com/SFxx_LTM_Tool/SBLTM-Linux-x86-64-LATEST.tar.gz

7.2 Swissbit Life Time Monitoring Library for Windows & Linux

Swissbit offers a software library (API) which allows customers to enable their applications to monitor Swissbit in detail device status.

Advantages of the SBLTM software library:

- Simple programming interface (C library)
- Hides the complexity of the operating system API (enumeration of the drives, identification and SMART commands)
- Hides the complexity of the S.M.A.R.T. structures, parsing and validation is done in the library
- Identical API for all supported operating systems and device types

The library is available for a small one-time fee per platform (distribution is royalty free).

Please contact your Swissbit sales representative for a quote or more technical details.

7.3 SD / MMC / microSD: S-2xo, M-100

For this product group there are vendor commands available for reading out bad block and spare block information and wear leveling (erase count) information. As the implementation of vendor commands does not follow the SD conventions, full control of the interface is necessary. As the implementation depends on the interface hardware, Swissbit does not currently supply a tool for this product group.

The documentation of the vendor commands is available under NDA. Please contact Swissbit (industrial@swissbit.com) for more information.

7.4 Life time assessment

We recommend assessing the expected/guaranteed life time of Flash based devices to make sure the product life time fulfills your expectations. The real application should be tested in a test installation or simulation for best accuracy.

In general calculations based on application behavior statistics are not possible, as the influence of the operating systems (with the caches) and the file system have a big influence on the data actually written on the device.

Many product groups report the real erase cycles that the NAND Flash blocks have seen. Based on the average erase count and its ascend in the target system, it is possible to quite simply calculate the guaranteed life time of the product. Please consult the data sheet for guaranteed life time information. Typical SLC NAND Flash currently guarantees at least 100'000 erase cycles per block.

For other products a rough life time assessment can be made with statistical data of write accesses on the interface (e.g. IDE / USB / ...). This method is more complex, may need special equipment, and often can't factor in optimizations of the controller in the calculations.

Bad and spare block counts can't be used for linear life time calculations. In the beginning of the device life time, only very few blocks will be needed to be replaced; only near the end of the life time more bad blocks will occur.

8 Glossary

8.1 NAND FLASH

Type of Flash typically used for data storage applications as opposed to NOR Flash used in embedded systems, typically for program storage. Gets its name from the NAND Cell structure used.

- PBA (Physical Block Address)
  - Actual physical address of Flash block the sector resides in.
- BLOCK
  - Inside the Flash the largest unit of data storage. Typically 128K or 256K for large block Flash. Currently an 8Gbit SLC Flash has 8192 Blocks
- PAGE
  - Inside the Flash device, equal to 4 (2K Page) or 8 sectors (4K page)
- Page Overhead Area
  - Each Flash page has extra bytes (currently 16) per 512 byte sector. This is typically used by the controller for overhead items such as CRC, parity, logical block number etc. This brings the physical sector size to 528 bytes.
- Flash Internal Copy Page (Copy-back)
  - A mechanism where data from one Flash block is copied to another without leaving the Flash and without Controller intervention.
- Two Plane Flash and Commands
  - The newer Flash has its data blocks divided into odd and even planes. There are two plane commands with such Flash, that allows simultaneous operations on one block from each plane. So there is a two-plane program, erase etc. Using these commands allows significant increase in speed. Currently most Flash has this feature.
- Flash Cache Program
  - This allows writing of new data while the previous data is being programmed into the Flash array.
- SLC (Single Level Cell) FLASH – Expensive
  - 1 bit per Cell.
  - Reliable with 100K typical Program/Erase Cycles.
  - Data Retention is 10 years.
  - Good for Solid State Disk (SSD) applications.
  - An ECC of 1 bit (in 512 Bytes) is generally recommended.
- MLC (Multi-Level Cell) FLASH – Cheap
  - 2 or more bits per cell; allows larger density in the same package.
  - 10K program/Erase Cycles before EOL.
  - Good for commodity applications
  - 4 bits (in 512 Bytes) or more of ECC
  - As the process shrinks below 50 nm, the expectations are that 12 bit ECC will be required.
- SDP, DDP, QDP Flash
  - Single Die package or Mono Die, Dual Die Package, Quad Die Package
  - SDP and DDP packages typically have 1 Chip Enable
  - QDP packages typically have 2 chip enables
- NAND Flash Manufacturers
  - Samsung, Hynix, Micron, Intel, Toshiba, etc.

8.2 Firmware / Controller Features

- ECC (Error Correcting Code)
  - NAND Flash requires the use of an ECC. Parity bits are added so that data errors can be detected and corrected, within the limits of the ECC used.
- CRC (Cyclical Redundancy Check)
  - Used for the Ultra DMA protocol to check the validity of the data. Also used to backup the ECC.
- Wear Leveling
  - Since NAND Flash has a limited life, its important that all parts of the Flash experience similar usage. Wear Leveling assures that.
- Bad Block Remap
  - The process of replacing a Bad Block with a good one from the spare block pool.
- LBA (logical block address)
  - This term defines the addressing of the device as being by the linear logical mapping of sectors
- DFA (Direct FLASH Access)
  - A means of data transfer between the controller and the Flash without intervention from the controller.
- Interleave
  - A method that uses the busy time from a Flash to device(s) to simultaneously send program/erase commands to another Flash device(s)
- Interleave Factor
  - The number of Flash devices that are simultaneously serviced for Program/Erases. For the F3 this is 1,2 or 4. Interleaving significantly increases write speeds.
- Anchor Block
  - Chip 0 Block 0. All permanent information stored here as well as operating firmware. It is the "Anchor" point for administrative information. The Anchor Sector is Sector 0 of the Anchor Block.
- S.M.A.R.T (Self-Monitoring, Analysis, and Reporting Technology)
  - A monitoring system to detect and report on various indicators of reliability, used to anticipate impending failures.
- NCQ (Native Command Queuing)
  - An extension of the Serial ATA protocol, where multiple write or read commands are queued in the devices to improve performance.
- TRIM (Data Set Management)
  - TRIM commands allow an operating system to inform storage drives which blocks of data are no longer considered in use and can be wiped internally.

8.3 Products / Interfaces

- Universal Serial Bus (USB)
  - Universal Serial Bus (USB) is a serial bus standard to interface devices to a host computer. USB was designed to allow many peripherals to be connected using a single standardized interface socket and to improve the plug-and-play capabilities by allowing hot swapping, that is, by allowing devices to be connected and disconnected without rebooting the computer or turning off the device. Other convenient features include providing power to low-consumption devices without the need for an external power supply and allowing many devices to be used without requiring manufacturer specific, individual device drivers to be installed.
    http://en.wikipedia.org/wiki/USB
- USB Flash Drive (UFD)
  - A USB Flash drive consists of a NAND-type Flash memory data storage device integrated with a USB (universal serial bus) interface. USB Flash drives are typically removable and rewritable, much shorter than a floppy disk (1 to 4 inches or 2.5 to 10 cm), and weigh less than 2 ounces (56 g). Storage capacities typically range from 16 MB to 64 GB with steady improvements in size and price per gigabyte. Some allow more then 100 thousand write or erase cycles and have 10-year data retention, connected by USB 1.1 or USB 2.0.
    http://en.wikipedia.org/wiki/Usb_flash_drive

- COMPACT FLASH (CF)
  - 50 Pin Parallel Interface Card, a subset of PCMCIA. Can operate in PC Card modes or TRUE IDE mode (like a Disk Drive). Typically used in professional digital Cameras
- ATA / PATA / IDE
  - AT Attachment Standard – ATA defines the physical, electrical, transport, and command protocols for the internal attachment of storage devices. A parallel Disk Drive Interface standard often referred to as PATA and IDE.
- Serial ATA (SATA)
  - Serial version of ATA. Capable of data transfer rates > 200 MB/S
- Secure Digital (SD)
  - Serial Interface memory card with Content Protection available with CPRM keys. Typically 4 bits, extended to 8 bits. Royalties.
- Multi-Media (MMC)
  - Similar to SD without any royalties. 8 bit capability

8.4 Transfer Modes

- SPI Mode
  - The SPI mode consists of a secondary communication protocol that is offered by Flash-based SD Memory Cards. This mode is a subset of the SD Memory Card protocol, designed to communicate with a SPI channel, commonly found in Motorola's (and lately a few other vendors') microcontrollers.
- PIO modes
  - Parallel Input/Output. Basic ATA/IDE data transfer mode using I/O methods. Slower than DMA modes. PIO0 – PIO6 are defined.
- DMA (Direct Memory Access)
  - A means of data transfer between device and host memory without processor intervention
  - Multi-Word (DMA) Modes: Basic DMA transfer modes for IDE drives. Allows transfers of multiple data words without processor intervention. Often abbreviated MDMA.
  - Ultra DMA (UDMA) Modes: Faster mode of DMA using double edge clocking. UDMA5. Speeds of 100 MB/S are possible in UDMA5 mode, 133MB/S for UDMA6 mode. CRC is included in the data stream.