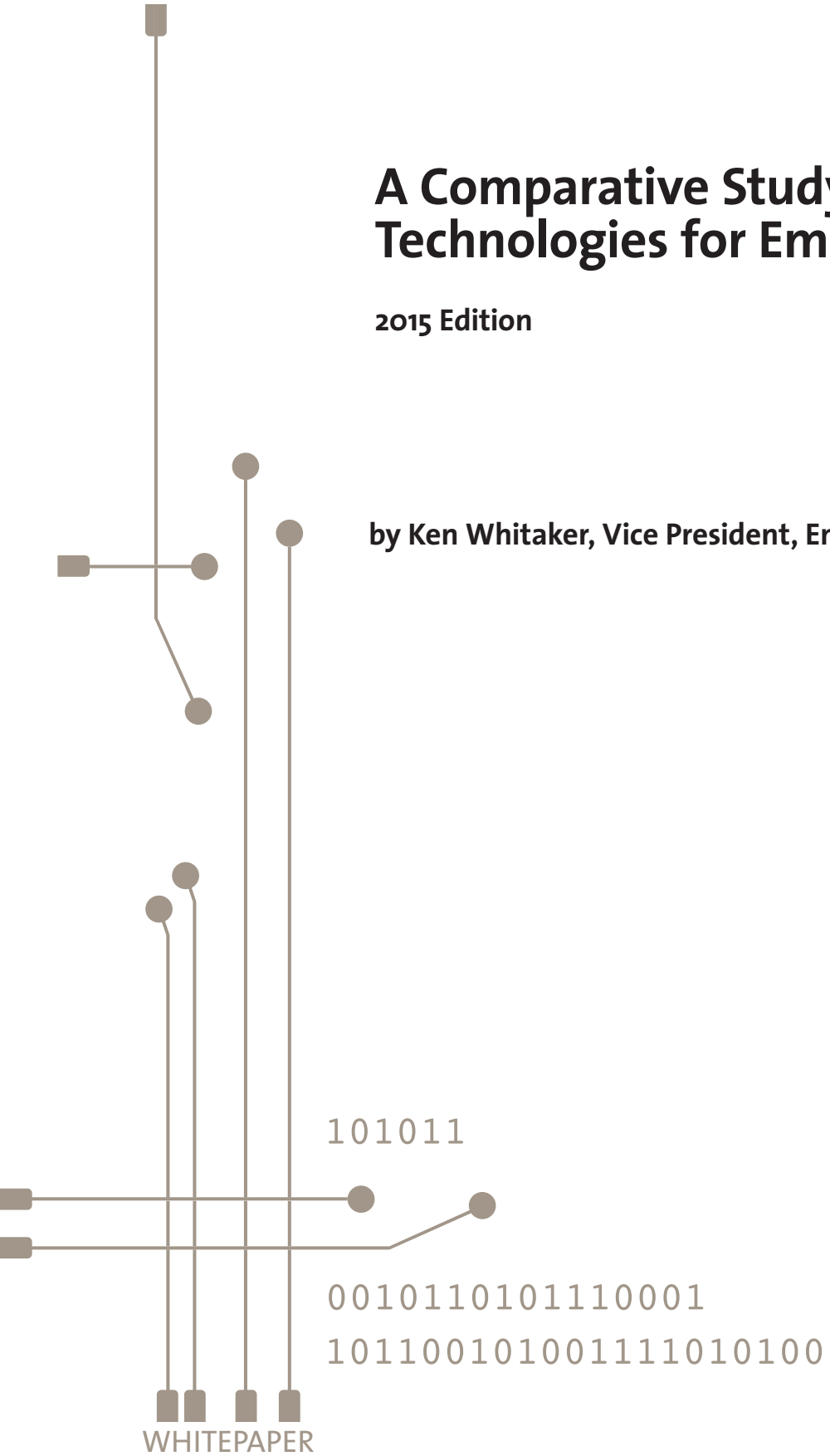
Datalight

A Comparative Study of Flash Storage Technologies for Embedded Devices

2015 Edition

by Ken Whitaker, Vice President, Engineering



Abstract

As mobile and embedded products become smaller, thinner, lighter, and more powerful, flash memory has replaced traditional rotating magnetic disk drives as the preferred storage media. As device manufacturers move increasingly to flash-based systems, they have an overwhelming variety of flash parts to choose from, each with its own unique blend of benefits and liabilities.

So how do you choose between different storage technologies for your embedded project? And what impact do other system components (operating system, device driver, and even applications) have on the end user experience?

In this paper we will identify what you need to know about flash memory and its surrounding ecosystem to determine what storage technology will best fit your next embedded project's data storage requirements.

Although flash memory has a huge enterprise and consumer presence, we will examine the current state of the art from a host software perspective primarily for the industrial user.

The first edition of "A Comparative Study of Flash Storage Technology for Embedded Devices" was published in 2012 and as flash memory standards change so quickly, it is Datalight's intention to update this report annually.

Topics

A Flash Memory Walkabout

Originally called *flash RAM* (random access memory), flash has rapidly supplanted the rotating magnetic disk. Dr. Fujio Masuoka invented flash memory in 1984, while working at Toshiba. [1] His colleague, Mr. Shoji Ariizumi, came up with the name *flash* because the process of erasing data from the cells reminded him of a camera flash.

That same year, Dr. Masuoka introduced this new technology at an IEEE meeting in San Jose, California. Intel immediately saw the benefits as an opportunity and released the first commercial flash memory chip in 1988.

Flash memory is a form of non-volatile memory that is electrically programmed and erased. The original electrically erasable programmable read-only memory (EEPROM) devices consisted of a grid of columns and rows with transistors at each intersection. A tunneling mechanism is used to place charge on the floating gate which affects the ability of a control gate to place the transistor in its conductive state. To change the value of a cell set to 0, a process called Fowler-Nordheim tunneling occurs changing the charge level in the cell. When power is removed, the cell retains its last setting after being programmed or erased. [2]

Modern-day flash memory operation has changed. Whereas EEPROM devices requires complete erasure before programming, NAND flash memory can be written and read in blocks or pages and NOR flash memory allows a single byte to be written or read. The technologies are different enough in that NAND has much higher capacity than NOR. *More on these technologies later …*

In any system where the form factor is constrained and permanent data storage is required, flash memory is frequently used. Portable devices like smart phones, data collectors, vehicle control systems, and tablets are the primary drivers of demand. This is especially true with consumer mobile devices.

The rapid evolution of flash-based, portable MP3 players gave way to powerful mobile smart phones and tablet computing devices that have become market forces driving flash memory adoption. Starting in 2011, Apple was one of the first personal computer companies to exclusively use solid state drives (SSD) on their popular MacBook Air instead of lower-cost hard disk drives (HDD). As of 2015, Apple offers flash-based storage across its entire Mac laptop and desktop computer product lines. Other laptop vendors are following the same trend as mobility, weight, and reliability become requirements to remain competitive.

In 2015, iSuppli, now a part of IHS Technology, has seen a slowdown in HDD worldwide production and is forecasting a rapid rise in SDD shipments (figure 1). [3]

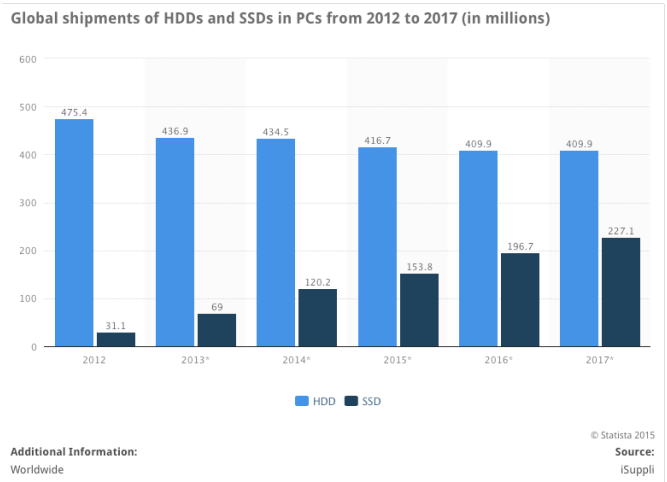


Figure 1: Global shipments of HDDs and SSDs in PCs in millions of units (larger values are better)

Flash in SSD form is also gaining traction in non-mobile applications such as computers acting as local servers and enterprise server farms, due to their requirements for performance, reliability, and low power consumption. The combination of flash memory with HDD as a caching scheme has also gained some acceptance in the category of solid state hybrid drives (SSHD).

By integrating a small amount of NAND flash (acting as a data cache) with a traditional hard drive, SSHDs provide the performance benefits of an SSD, the storage capacity of an HDD, and a price-point just slightly higher than that of a traditional drive. According to Seagate, SSHDs are up to 4.5 times faster than comparable 7,200 revolutions per minute (RPM) hard drives, with much higher capacities than available on SSDs. [4]

Benefits and Liabilities of Flash

Hard disks still command much lower prices, costs, and offer higher capacity, but when it comes to footprint (size) and lack of moving parts, both key for mobile devices, flash has become the clear winner. The miniaturization, density, capacity, and ruggedness of the flash chip are key drivers behind flash memory's meteoric adoption rate. But flash, like any technology, comes with both good and bad operational behavior—although as its growth confirms, the good by far outweighs the bad.

The Good

Speed, speed, and more speed: Flash memory chips have no moving parts and the more that data can be grouped together for optimum reads and writes, the better.

Very small, and highly dense footprint: Permanent storage can be squeezed into a very small space as the latest generation of USB flash drives are not much bigger than the USB connector itself. Wearables and Internet of Things (IoT) devices that require local permanent flash storage don't usually have much board space.

"Green": Low power consumption and little to no heat dissipation is rapidly becoming the norm.

Compatibility: Applications can use flash without special software since the flash memory subsystem can emulate traditional HDD without the risk of mechanical failure.

Reliability: Without moving parts found in rotating media, device reliability can be dramatically better, and, since there's no physical head movement, random performance should improve as well.

The Bad

Wear: Flash, unlike traditional rotating media, is susceptible to wear. As flash memory lithography becomes even smaller, flash lifespan limitations must be managed more aggressively.

Obsolescence: As flash technology is constantly moving forward, part obsolescence can have a huge impact on OEM production cycles.

"Neighbors behaving badly": Similar to HDD behavior, individual flash memory cells are susceptible to bit disturbance and failure. This requires that the controller or flash software provides sophisticated error detection and correction algorithms.

Data loss: Loss of power at the wrong time and without proper protection can result in incomplete writes that can produce erratic behavior. This requires protection responsibility in both the hardware design and software system so that flash memory is programmed to specification. This is a problem that is not unique to flash—this is a problem on rotating magnetic media as well. The file allocation table (FAT) file system wasn't reliable in power interruption situations on disk drives long before flash memory.

Lack of security: In applications where deleted data must be wiped with no trace, flash memory's ability to assure complete erasure is in its infancy. [5]

Quality: Differences in chip fabrication processes can impact the quality, timing, and even performance of flash parts, even in chips with the same manufacturer or part number.

Before diving deeper into the technical differences between flash technologies, let's first look at flash memory from the end-user perspective.

What the End User Expects

Ultimately, it is the experience of end users that will make or break a flash device manufacturer. Flash memory that works *most of the time* creates a nightmare of customer returns for OEMs and system integrators. In an environment of continued market competition, failure to satisfy the end-user usually results in a commercial product failure.

As you assess the data storage needs of a typical end user, you may want to rank your project requirements in a chart like this one, with prioritized attributes based on the target market's use case and user expectations.

Priority	Industrial	Consumer	Enterprise
1	Reliability	Cost	Security
2	Performance	Performance	Performance
3	Endurance	Security	Reliability
4	Security	Reliability	Power consumption
5	Power consumption	Endurance	Endurance

Table 1: What end users expect of data storage attributes by industry (industrial market highlighted)

Let's examine the key attributes in more detail.

Endurance

Whenever data is written, users naturally expect it to be available for later reading. Flash memory can last for years or, depending on which part of the flash is repeatedly read, erased, or written, it could wear out in a matter of days. Older SLC flash memory should be able to sustain at least 100,000 erases while other flash memory, like multi-level cell (MLC) flash, can only be erased 5,000 times or less. Triple-level cell (TLC) is even worse with erase cycles of less than a few hundred times due to the 1X and 1Y lithography. The use of sophisticated wear-leveling algorithms in controlling software can help to mitigate this phenomenon; however, as densities continue to increase and dies shrink, endurance is only going to get worse.

Figure 2 shows that the more dense the flash cells are, the less number of times erases and writes can be performed before permanent cell failure. [6]

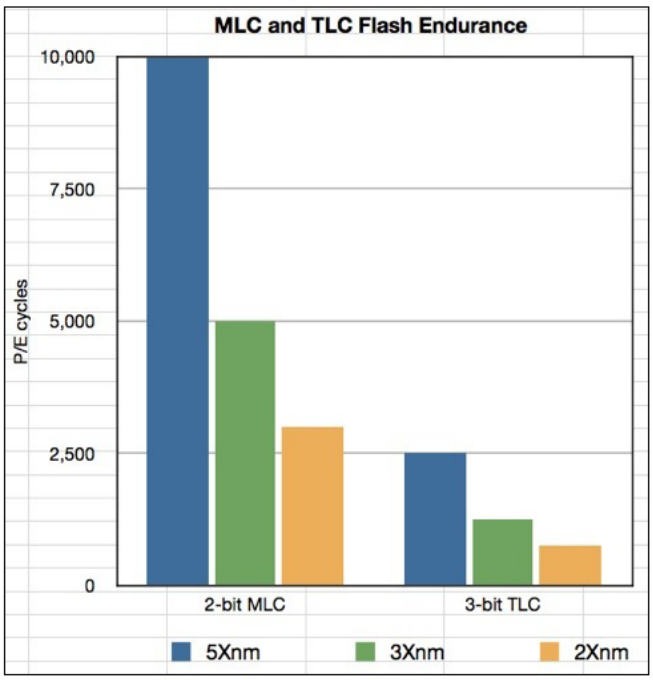


Figure 2: Comparison of MLC and TLC Endurance shown in program/erase cycles (larger values are better)

For flash-based devices with an expected short lifespan (a throw-away phone, for example), endurance is less of

an issue; however, for ruggedized handheld terminals, meter reading devices, or automotive systems, customers will expect these devices to last more than ten years. Thus, understanding your customer's endurance needs is vital to selecting the best flash technology.

Performance

Performance requirements may be expressed in terms of raw throughput, input/output operations-per-second (IOPS), latency, mount time, boot time, and other measurements. Attempting to "be all things to all users" by creating the best performance for every use case is a recipe for disaster, if not unattainable.

Relying on standard benchmarks may not tell the whole performance story, quite possibly giving you a false sense of acceptable performance. The results from your benchmark tests may not come close to the actual customer-specific application performance. Most customers who demand peak performance will undoubtedly run a slew of their own benchmarks, with the expectation of their entire solution beating their competition.

There are a few key performance metrics you should be familiar with.

Throughput: *The definition of throughput depends on the application. For example, your customer may require fast streaming (in other words, sequential read throughput) for music and movie playback. New flash memory will typically perform dramatically better once data has been written and rewritten causing fragmentation of used and available blocks.*

Other customers may expect fast throughput on random database access for their customer relationship management (CRM) application—especially true for Android embedded systems that relies on SQLite for many of its file-based operations.

SQLite's dependency on random file requests depends on the flash technology employed. For example, embedded multimedia memory card (eMMC) flash excels on sequential reads, but suffers with random record read/write requests. Advances in page-based eMMC is making this less of an issue, however.

IOPS: *This is a common measurement used to determine the number of requests passed onto the flash device from the host computer. IOPS results can be misinterpreted based on a number of factors including the following:*

1. Theoretical IOPS numbers published by flash vendors may not represent the real world where the real-time operating system (RTOS) or other hardware components will impact actual system performance. This theoretical performance depends on a number of factors and places pressure on host flash software providers to move data at unrealistic published performance numbers.
2. Sequential operations access locations on flash memory in a contiguous manner utilizing large record sizes will generally outperform random operations that access flash memory in a non-contiguous manner. These random operations, depending on the application, are typically used by database applications with smaller record sizes than that of sequential access applications.
3. In the old days, data storage operations were generally queued up one after the other; however, a mainstream industrial RTOS incorporates multithreading where the number of threads and queued up data requests appears to run in parallel.

   There is a performance cost to managing multithreading, but the benefit is that tasks (and, ultimately, the end user) won't appear to be hung waiting for file requests to complete. In conjunction with performance, safety critical standards like Avionics Application Standard Software Interface (AE653) require deterministic behavior for completion of file operations. [7]
4. The data block size matters since moving large chunks of data at a time should improve throughput (but reduce IOPS).

Latency*: Latency describes any delay between the point when an I/O request is issued and when the operation actually begins (usually in the form of data transmission). This is a known issue with rotating media due to head movement, but flash memory is not immune to it, especially when general flash housekeeping such as cache management, garbage collection, and reading entire blocks to find a particular page, needs to take place. [1]*

Mount time: *This usually refers to the time it takes for a file system to be ready for its first operation. Mount time is usually expressed in seconds; smaller is obviously preferred.*

Mount time varies widely depending on which file system is used. For example, the journaling flash file system

(JFFS2), the unsorted block image file system (UBIFS) file system, and Samsung's flash-friendly file system (F2FS) are relatively slow to mount and becomes even slower on larger flash sizes JFFS2, like many other file systems, maintains a historical log of file and folder activities. Scanning the log to rebuild the file system's directory structure can also be very slow, depending on how the system was previously shut down. [8] UBIFS is a successor to JFFS2 and works with raw flash. UBIFS is also slow during mount, but there is a proposal for future development that maintains bad block tables in flash to avoid scanning all physical blocks at boot time. Datalight's own Reliance Nitro file system uses a patented method for high reliability and doesn't rely on journaling or any other cumbersome mount time validation.

Boot time: *Boot time is a measurement of mount time, plus the time required to read necessary data such as the OS kernel, initialization programs and configuration files for all file system volumes. Poor boot time performance can significantly impact an end user's startup experience. This can be especially important to users of mobile embedded systems like radios and handheld industrial devices. A mobile phone user, for example, may be inconvenienced by slow boot time, but the job of a fire fighter or policeman demands instant on capability.*

It is important to break down the components of the boot sequence when diagnosing problems in this area. For example, a 40 second boot time may consist of 20 seconds for the RTOS to initialize, 18 seconds to applications initialization, and only 2 seconds for software components that work with flash memory. In this case, focusing on improving the flash management's boot time may not improve the overall user experience by much.

Boot time is usually expressed in seconds—smaller numbers are preferred. Fully understanding the customer's *use case* should be the first priority when benchmarking a system. Also, it's important to properly configure the hardware and software to accurately model the expected end user behavior.

Some of the most popular, non-proprietary benchmarking applications include:

- `IOzone`: Linux file system benchmark, although it has been ported for other operating systems.
- `Fio`: Linux Flexible I/O tester benchmarking tool provides a number of configurations to simulate true I/O loads.
- `Bonnie++`: Linux file system benchmarking application that focuses on data read and write speeds, number of seeks per second (mostly used for rotating disk media), and number of file metadata operations per second.

Most of these applications are available in source code format and can be ported to most any RTOS. Figure 3 shows sample benchmark results using `IOzone` on two file system devices configured with different block sizes.

Record Size

	4	8	16	32	64	128	256	512
FS A - 4k	3994	4197	3958	2826	3346	3323	3319	3359
FS A - 8k	2322	1683	1846	1777	1729	2028	1919	2159
FS B - 4k	2755	2727	2744	2720	2728	2738	2724	2726
FS B - 8k	2731	2747	2809	2770	2771	2773	2745	2751

All values in KB/sec

Figure 3: Comparison benchmarks with `IOzone` (larger values are better)

Power Consumption

Particularly in mobile industrial devices, where battery life is becoming a competitive advantage, power consumption is a top design priority. Flash memory and its associated adjacent circuitry is one of the power hungry electronic components in embedded design. Some designers are using cached memory buffers to limit the number of read/write requests, which can dramatically improve up time. Another method to save power is hibernation during limited or inactive use.

For write-intensive applications, NAND flash consumes significantly less power. Over time, the total energy used will be significantly greater for NOR. [9]

For those platforms that are not battery operated, like programmable logic controls (PLC) used in industrial factory automation settings, power consumption may not be a priority; whereas, for consumer mobile applications like smart phones or tablet computers, stretching battery even by a small percentage can be a huge competitive advantage.

Security

One of the most obscure and misunderstood flash issues

is the secure erase of data. Think you've erased all data on a block? You probably haven't. With all the focus on privacy and security for digital information, very little progress has actually been made in this area. It is the responsibility of the software to ensure that data is truly erased.

A typical flash-based system will tell you that data is erased when actually it has merely been deallocated. Most data is still recoverable until the block actually undergoes explicit erasure. Allocation schemes may also mask the fact that, even though erased data appears to be permanently removed from the media, it has actually only been moved.

In a 2010 study by the University of California, San Diego (UCSD), researchers took the time to examine the effectiveness of erasing SSDs at the file system level. The researchers examined a variety of scrubbing techniques in addition to the impact on performance as well. After evaluating flash parts from six different manufacturers, the study determined that only a comprehensive secure erase and overwrite of the entire flash media destroys data. Using software to perform file-by-file erases, however, does not work. [5]

In cooperation with Joint Electron Device Engineering Council (JEDEC), Datalight has provided file level support for eMMC secure operations that provides an effective means for securely erasing data on flash media.

As more sensitive proprietary data is being written to flash, data security is becoming a priority for manufacturers and content providers alike. With the Next Generation Secure Memory initiative (NSMi), [10] a joint project by SanDisk, Toshiba, Sony, Samsung, and Panasonic is delivering optimized memory solutions to enable data storage with robust security technologies that can protect premium content. Through the use of encryption and other techniques, they hope to safeguard access rights to data content regardless of the storage media.

In addition to working with secure data storage, NSMi is defining an alternative way of transferring data intuitively and seamlessly between devices and the cloud. [11]

Understanding Usage Patterns

Prioritizing the above items based on your use case is the first step to understanding which flash technology is right for your design; one flash technology can't possibly excel in all categories. Even similar technologies from different vendors will behave differently.

The number of use case variations is essentially limitless, but there are a few broad categories that we see in many industrial flash-based designs:

- Casual use, browser-centric
- Streaming data playback
- Streaming, real-time data download
- Database-centric random access
- Data collection for automotive, telematics, and handheld mobile collection devices

These categories are identified as a general guide to frame the discussion of differing flash characteristics, as the remainder of this whitepaper discusses benefits and liabilities in relation to these use cases.

Size and Packaging

Flash die sizes have been shrinking since the technology's invention. Manufacturers are producing parts in the sub-20nm range, which introduces new challenges in error correction and other limitations of the technology. The primary reasons for the constant miniaturization of flash are two-fold: financial (smaller die sizes contains more chips on a silicon wafer) and space in order to fit on increasingly smaller mobile and embedded industrial boards.

Multi-chip packages (MCP) have the advantage of saving board space by integrating several memory products into one package. Toshiba MCPs, for example, combine static random-access memory (SRAM), dynamic random-access memory (DRAM), and NAND flash into a complete memory subsystem using a chip-stacking technology. The primary benefit is size and cost reduction for small portable devices. [9]

Back in 2012, TDK announced a new embedded solid state disk (eSSD) MCP that consists of NAND flash and a controller chip in a single 17x17nm package. [12] According to TDK, this reduces the footprint by two-thirds

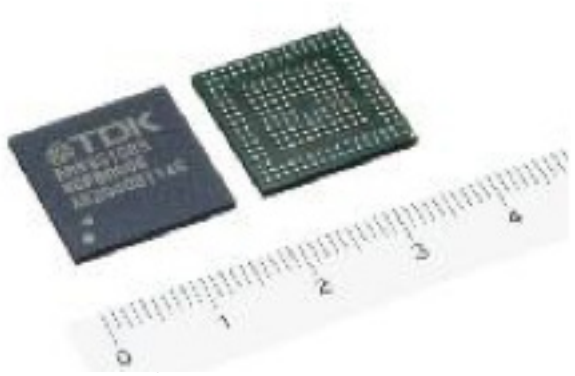


Figure 4: TDK's eSSD multi-chip package

This package uses a standard serial AT attachment (SATA) interface with write performance around 30 MB/sec and reads are around 55 MB/sec. Capacity is up to 4 GB. TDK describes the primary market for these new parts as commercial devices such as multifunction printers and point-of-sale devices with small footprint and high performance.

One of the newer players in the packaging options for flash memory is serial peripheral interface (SPI) flash. The rise of small IoT devices has provided the motivation for reduced-footprint packaging options. SPI flash is basically ordinary NOR or NAND flash wrapped in a small controller that operates on a SPI bus. The commands used to drive it have been augmented to control SPI functions such as the number of active I/O lines. And in the case of NOR flash, commands have been added to control flash erasure that ultimately reduces the complexity of the host software driver.

The use of a SPI controller allows the flash to be placed in small footprint packages with reduced pin outs—a primary requirement for IoT devices.

Flash Memory Comes in Many Flavors

So what are the fundamental differences between different flash technologies? Skipping the minutia of the substrate, chemical behavior and other details of the inner workings of flash, we'll highlight the basic categories and discuss differences in the principles of flash operation.

Raw Flash

At the most basic level, there are two types of flash commercially available: NOR and NAND.

- ***NOR:*** NOR flash was originally designed to replace read-only memory (ROM), erasable programmable read-only memory (EPROM), and EEPROM non-volatile memory devices. NOR has a full set of address lines, enabling byte access for reads. In fact, reads are typically as fast as DRAM, allowing programs to be run execute in place (XIP), directly from NOR memory.

Erases however, are very slow, which is not a problem, as long as they are performed infrequently. NOR was originally used in removable CompactFlash (CF) cards. Since NOR doesn't require host bad block management or ECC support, it is a good choice for storing operating system images, system boot loaders, and system configuration information.

Overall, NOR is best suited for lower-density, high-speed read applications. [9]

- ***NAND:*** NAND was developed to achieve higher capacity, smaller footprint and lower cost over NOR, but NAND is limited to a minimum programming size: the page. As a result, individual bytes cannot be directly accessed; meaning reads, writes and erases must be performed in chunks. This technology is an unsuitable replacement for ROM, but fits quite nicely as a replacement for hard disk drives. NAND has become the standard flash technology for removable media, such as CF, Secure Digital (SD), and xD-Picture Card. NAND is most ideal for applications requiring low-cost, high-density and fast erase. Table 2 shows key differences between the two technologies:

	NOR	NAND
Typical capacity	*128 MB*	*512 MB - 64 GB*
Endurance (maximum erase cycles)	*100,000 to 1M cycles for SLC NOR, up to 100,000 cycles for MLC NOR*	*2,000 erases for MLC NAND, upwards of 50,000 erases for SLC NAND*
Erase	*Block*	*Block*
Program	*Byte*	*Page*
Default state	*Each bit set to 1*	*Each bit set to 1*
Access	*Byte (random access)*	*Page*
Programming time	*Slow*	*Fast*
Read time	*Fast*	*Depends*

	NOR	NAND
Uses	*Replace ROM or EPROM for program storage*	*Most USB, memory cards, SSDs*
Quality	*No errors, fault free*	*May have bad blocks*

Table 2: Comparison between typical NOR and NAND

Recently, some hardware engineers have been designing boards with NAND replacing NOR (or some combination of both) and, in some cases, these different technologies are being packaged in a single MCP.

Table 2 introduces the concept of *blocks* and *pages* used in erasing and programming flash. Each block consists of a number of pages where a page size typically is either small block (512 bytes) or large block (2048 bytes, 4096, and so on). The block size is important because you'll typically get the best results if record sizes are *aligned* to an even fraction of the underlying block size used on the flash media. Figure 5 is a visual representation of the strengths and weaknesses of NOR and NAND. [8]

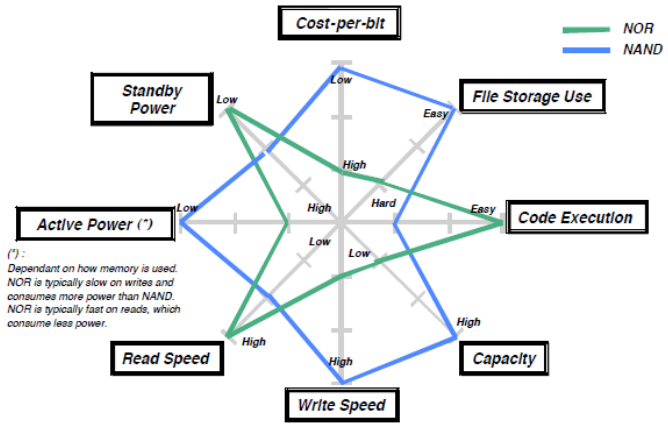


Figure 5: Comparing NAND and NOR flash

Where NOR excels in code execution, read speed, and standby power, NAND excels in cost-per-bit, file storage, capacity, write speed, and active power consumption (table 3).

	NOR	NAND
Cost per bit		✓
File storage		✓
Code execution	✓	
Small host driver (memory footprint)	✓	
Capacity		✓
Write speed		✓
Read speed	✓	
Erase speed		✓
Active power consumption		✓
Standby power	✓	

Table 3: Checklist between NOR and NAND

SLC versus MLC

Most flash memory is organized into a grid of memory cells (floating-gate transistors to be exact), with each cell storing a single bit of information. This is called single-level cell (SLC), and it is still in use today in the most demanding applications.

To increase storage capacity and to reduce costs, flash manufacturers developed MLC devices, which store more than one bit per cell. MLC has evolved from two bits to three bits per cell and, to further complicate matters, NOR and NAND can be either SLC or MLC. The choice depends on your customer's requirements for power, speed, quality, and endurance. [13]

Table 4 compares the strengths and weaknesses of SLC and MLC (and TLC) NAND.

	SLC	MLC/TLC
Bits/cell	*1*	*2-3*
Capacity		✓
Cost per bit		✓
Size (footprint and packaging)		✓
Best performance	✓	

	SLC	MLC/TLC
Endurance	✓	
Quality (limited error correction required)	✓	
Power consumption	✓	✓

Table 4: Comparing SLC versus MLC NAND

The write performance of MLC is typically about one-third to one-half as fast as SLC. Its increased complexity also creates more bit errors, requiring more sophisticated error correction code (ECC) to prevent data error conditions.

The table above may lead you to believe that SLC is a better technology than MLC, but for customers needing higher storage capacity at the lowest cost who can live with the inherent quality issues, MLC is actually the better choice.

As far back as 2007, about 80 percent of all NAND shipments were MLC. [13] In fact, MLC-based SSDs and USB flash drives may well be the ideal storage technology for personal computing. If the USB flash drive fails after only a year, replacement is a cheap and easy fix. SLC-based SSDs, on the other hand, might be the best choice for enterprise server use, where fast performance and high reliability are more important than cost-per-bit. SLC is definitely a great fit for industrial embedded applications.

However, the reduction in lithography size has blurred the lines somewhat between MLC and SLC. Now, some modern SLC devices have limited erase cycle ratings and large EDC.

Consider working with your flash vender to develop models to help determine which technology is right for your application. The work Toshiba has done is an example of a thorough study of customer usage habits. According to Toshiba's analysis of typical user data patterns, to reach the customer's expected endurance limits of a 64 GB MLC NAND-based SSD, a user would have to write 40 TB of data over a five-year period (roughly 22 GB per day).

In some cases, the choice between SLC and MLC is to actually use both. Just as hardware engineers often include both NOR and NAND on a single embedded board, SLC and MLC can be combined to provide a customized data storage solution as long as the proper partitioning tools are available.

Don't Forget about TLC

Always striving for higher density at lower cost, flash vendors have introduced TLC flash, resulting in even greater (three bits-per-cell) capacity over two bits-per-cell MLC. Back in 2010, Intel and Micron jointly announced a 25nm TLC NAND device of 64 Gb (8 GB) that was twenty percent smaller than a similar capacity 25nm MLC part. [14]

With the advantages of more capacity, lower power requirements and smaller size, TLC also has distinct disadvantages, including slower transfer speeds, higher error rates (requiring even greater ECC), and lower endurance than its SLC and MLC counterparts. Of course, TLC is a great candidate for consumer products where low cost storage trumps reliability and performance.

Along Came Managed NAND

In order for NAND to be usable, it needs a controller that connects the host computer to the flash and these days the CPU usually has built-in flash controller circuitry. [15]

Figure 6 shows the difference between typical NAND that is controlled by the host computer, and managed NAND where the control and advanced flash management features are now performed in a controller chip (containing control logic and firmware embedded in the part). [16]

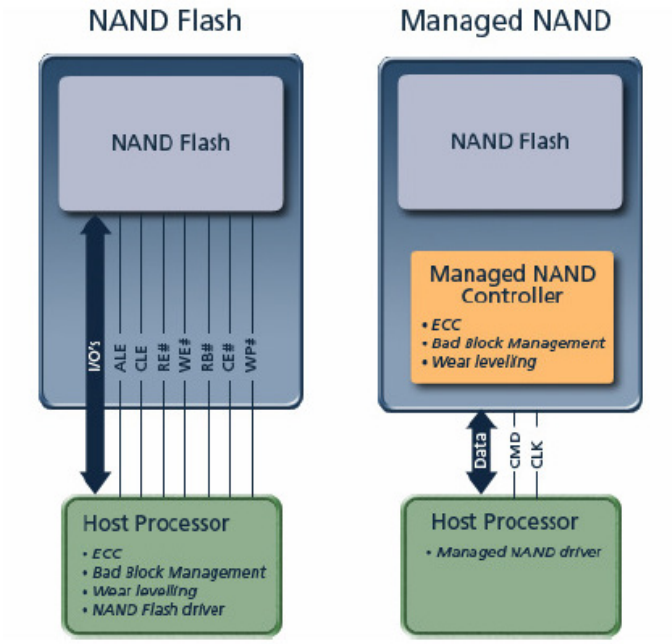


Figure 6: NAND flash versus Managed NAND

Because modern NAND requires an increasing amount of management by the host computer and associated flash management software, device designers have demanded more efficient approaches. ECC-free NAND devices help make technology transitions nearly seamless by handling media management and error correction code (ECC) internally.

Managed NAND incorporates a controller with flash memory into a single package that has dedicated flash translation layer (FTL) firmware to perform functions like ECC, bad block management, and wear leveling so that the host computer software doesn't have to. This effectively enables the flash to behave like a typical block device, a simpler interface for the file system. The management functions performed by the integrated firmware are generally not visible to the host computer or its software, further simplifying the system.

Note: Some NAND flash parts now incorporate some advanced features that can positively impact host software performance. An example is Micron M60 NAND flash that offers built-in ECC handling.

Over the past couple of years, eMMC has become sought after in consumer products that need the capacity of MLC and the simplicity of managed NAND.

According to IHS Technology, worldwide shipment of eMMC parts are growing as much as 40 percent per year. Back in 2012 our edition of this whitepaper stated that 500 million units of eMMC was forecasted to ship worldwide. The actual 2012 shipments was well over 650 million units. According to IHS Technology, over two million units should be shipped by 2017. [17]

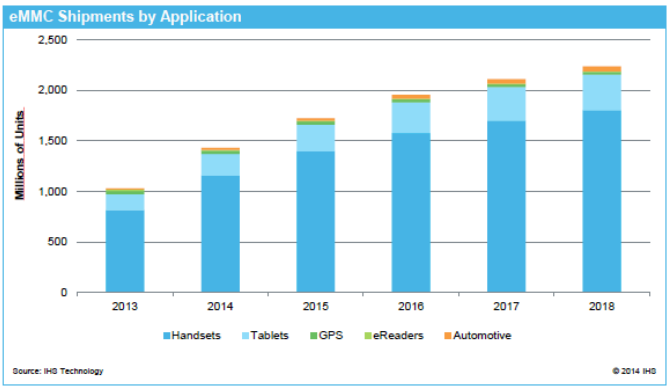


Figure 7: Worldwide shipments of eMMC

The standards body responsible for eMMC, known as JEDEC, is putting a lot of emphasis on enhancing eMMC standards. For the latest approved eMMC 5.1 version, the focus has been on improving reliability and performance. [18]

Performance improvements include:

- Interface bandwidth should increase to 400 MB/sec (from 104 MB/sec).
- Packed commands allow a series of commands to be grouped into a single data transaction.
- A context ID where different memory transactions can be grouped under a single ID, informing the device of the existence of related data.
- Data tags allow specific write transactions to be identified and prioritized for a specific memory region to improve performance and reliability.

Reliability improvements include:

- Power-off notification ensures data reliability when power is removed.
- Dynamic device capacity provides a way for the device to continue operating (with reduced total capacity) when erase blocks wear out. Host real-time clock information can be retrieved to help the controller's logic perform tasks such as improving internal memory management.

Another benefit of eMMC 5.1 is enhanced partitioning, allowing partitions to be designated for different memory types. For example, one partition could be created on faster, more reliable SLC for a file system's metadata, and another created on slower, less reliable integrated MLC for user data. Terms like *less reliable* are relative, and hopefully the controller and its firmware can remap data from bad blocks to good blocks without the host system even knowing.

Host software RTOS environments like Android are requiring more non-volatile memory (currently 4 GB), and capacity continues to grow with the availability of low-cost applications. eMMC offers a tremendous opportunity for portable devices such as tablets and smart phones to offer generous storage in a very small footprint. [19]

Though it represents a great leap forward in taming the idiosyncrasies of NAND, eMMC is not a perfect technology. After running a series of performance benchmarks on eMMC, Datalight's engineering team observed lackluster performance, particularly in random reads on files larger than 16 MB. In addition, figure 8 shows a sample test comparing different record sizes between raw NAND flash and eMMC on an ARM embedded target.

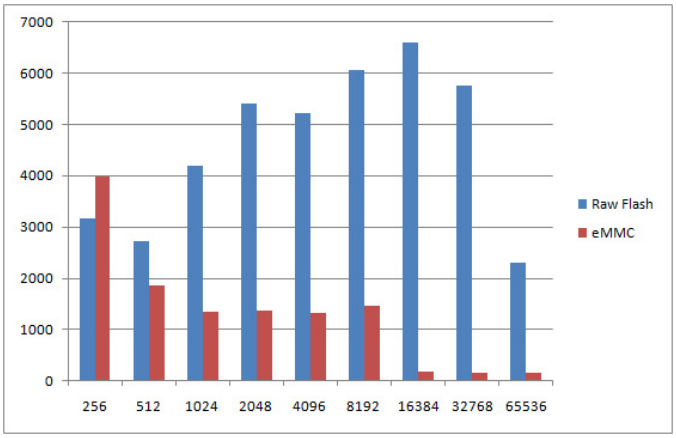


Figure 8: A sample test run of random read performance of eMMC versus NAND in KB/sec (larger values are better)

Although performance differs among specific vendor eMMC parts, we found poor random read performance to be consistent among manufacturers. For example, if your design will be running Android with applications executing many small record requests (using the built-in SQLite database engine), eMMC may not give your customers the user experience they expect.

One additional note of caution about eMMC: hardware innovations are advancing much faster than host software can keep up. Without software support, many of these innovations are unavailable to both OEMs and users. The concern heard from flash manufacturers is, "We keep adding great features, but software support is nowhere in sight!"

Move Over, eMMC—Here Comes UFS!

As shown in figure 7, eMMC shipment rates are projected to level off by 2016, typical when existing technology is being replaced by a better one. In this case, the battle is between eMMC and a newer JEDEC specification called universal flash storage (UFS). UFS samples are now becoming available, but there is consensus by founding UFS members (Micron, Nokia, Qualcomm, Samsung Electronics, and Toshiba) that a high performance, more secure, low power consumption device is needed.

UFS relies on the industry standard MIPI Alliance M-PHY and UniPro specification that uses a high-speed serial interface coupled with SCSI command protocols, which enable multi-threaded, concurrent requests. UFS was slated to support throughput of over 350 MB/sec, and the addition of command queuing, random read and write speeds, should be dramatically better than eMMC. [20]

With UFS 2.0, advances in flash technology promises interface speeds of up to 1.2 GB/sec, which is three times faster than the most cutting edge eMMC 5.1. [21]

It also looks as though UFS will embrace consumer industry hardware and software standards better than eMMC, thereby enabling software developers to streamline integration and reducing the investment required by OEMs.

Quirks and Challenges of Flash Memory

The speed at which flash memory has taken over in so many markets is a testament to its superior ruggedness, performance, and scalability, but in some ways it is still less than optimal. Flash quirks like the need to block erase and issues handling bad blocks, wear leveling, write amplification, and a host of other challenges keep software developers and OEMs busy mitigating the shortcomings of flash memory.

You Must Erase Before Writing

Unlike rotating media (hard disks), flash memory requires a separate and sometimes time-consuming erase of data in order to turn all bits back to the original 1 (on) state. [9]

Once a bit has been programmed to the 0 (off) state, it cannot be changed back to a 1 without being first erased. Thus, erasing sets all bits to 1, and programming clears selected bits to 0. [21]

Another flash quirk that makes it dramatically different from rotating media is the way erases are performed. Flash does not allow individual bytes to be erased, and can only be erased in large chunks (blocks). Erase blocks are defined by the manufacturer, typically some multiple of pages. So reading and writing are performed per page and erasing is performed per block, where a block is usually a power of two multiple of pages based on the block size. For example, a 16 KB block is composed of 32 512-byte pages. For a 2 KB page, the block may be 128 KB (composed of 64 pages). The reason why erasing happens in large chunks is for throughput and performance. Erasing is a much slower operation than writing, and if erases took place in smaller sizes, drivers would quickly bottleneck waiting for erases to complete.

This, combined with the limited erase cycle rating, introduces what may be the biggest difference between rotating magnetic media and flash: individual sectors cannot be overwritten in place with flash media. A sector of data to be updated must instead be copied and, as a result, the existence of stale copies of sectors is one of the largest challenges. Not being able to overwrite old data can be an issue with security.

To get the exact block/page configuration for a particular flash part, you'll need to check the manufacturer's datasheet. [22]

Block erases can increase wear on flash parts, but the limited lifespan is a function of the underlying physics of the CMOS transistors that make up the storage array. A sophisticated file and flash management system can organize data and metadata in ways that minimizes the number of erases, thereby extending the life of the flash. This is especially important as flash lithography shrinks, and with it the maximum erase count for flash parts.

Extending Flash Life With Wear Leveling and Bad Block Management

Because flash memory has a finite lifespan, rewriting the same block continuously will cause flash parts to wear out prematurely. To prevent this, wear leveling algorithms were invented, spreading wear evenly across the flash. FTL software makes sure that write requests are distributed to less frequently erased blocks. Poorly designed file system and FTL software can essentially destroy good flash memory. For example, standard FAT file systems continually write and rewrite the file allocation tables at the beginning of the media. If the FTL doesn't continually relocate these blocks, these writes could quickly wear out the flash.

It is important for wear-leveling algorithms to successfully balance erase cycles across the flash. Redistributing and remapping data on the flash accomplish wear leveling, but this process itself will introduce erase cycles. If done too aggressively the wear-leveling algorithm will itself become a burden on the device erase-cycle rating.

As individual blocks wear out over time (or are sometimes bad from the factory), bad block management (BBM) software is another important part of the FTL. When an attempted program or erase operation has failed, BBM goes into action by remapping the bad block to a spare one reserved for bad block allocation. [18]

Minimizing Write Amplification

Write amplification is described by the ratio of the size of requested writes to how much data the FTL must move around in order to accommodate it. Block-based FTLs must often move and rewrite entire erase blocks of data to accommodate a single-page write, especially in random write scenarios where the FTL cannot keep many erase blocks actively open, and must perform erase-and-copy operations to close them. This effectively forces the FTL to copy extra data when writing a single block.

Because flash uses metadata to track the physical location of data, and because data is typically smaller than the page it is written to, flash-based systems can take additional time and disk space when writing any file. Some advanced file systems are able to control the number of erase and program operations to individual blocks, with the goal to keep data evenly distributed across blocks.

Wear leveling works by relocating and remapping data, but with a properly designed wear-leveling algorithm

this actually doesn't have to happen very often. Opening and closing erase zones for writing, on the other hand, happens all the time, especially in random-write regimes.

This data movement assumes that the system is mapping logical blocks to physical blocks. For example, a request from the host to access block 20 may initially access that physical block, but may later map and access a completely different block.

Consequently, the flash host software needs to minimize write amplification where a write won't cause many more writes. Remapping data to facilitate wear leveling can unfortunately trigger additional erase and program cycles that increase the effects of write amplification. [16]

Program Disturb "Jitters"

The unintentional programming of an unselected (or inhibited) non-volatile storage element during a program operation that intends to program another non-volatile storage element is referred to as *program disturb*.

There are several methods flash designers can use to diminish the effects of program disturb, such as programming pages in a block sequentially (starting at 0 to page 63 on SLC), minimizing partial page programming (on SLC), and restricting page programming to a single operation. [3]

In contrast to program disturb, there's the phenomenon of read disturbs where a read of flash causes nearby cells in the same memory block to change over time. This situation usually occurs after hundreds of thousands of reads between erase operations.

But It Was Just There a Minute Ago!

In addition to making sure that the surrounding hardware components properly enable flash operation, there is one important function that is often overlooked: ensuring that write (and erase) operations actually complete.

Interrupting the programming of a NAND page by power loss, assertion of the write protect (WP) signal, or issuing a reset command can result in problems that are hard to detect or work around. A partially programmed NAND cell may not consistently read back the same value, or may have poor data retention. Historically, NAND chip manufacturers have tended to ignore these problems in their specifications, but recently they have become more explicit in prohibiting this.

Data Retention

Data retention is an evolving characteristic with little long-term data to support just how long flash can effectively maintain its data.

Even after data has been successfully programmed to the flash, there is a concern over data retention. Since SLC has much larger program/erase cycles compared to MLC, SLC should have longer data retention than MLC. That is why enterprise servers typically rely on SLC flash. Keeping the flash in active use with constant refreshing of data should improve the likelihood of long-term data retention. [23]

Bit Errors and ECC

Each NAND page has additional bytes set aside in the redundant area (also known as the *spare area*) that are typically used store error correcting code (ECC) checksums. Error detection is the logic used to detect read-bit failures. Detection and correction go hand in hand, and common algorithms include Hamming (SLC), Reed Solomon (MLC), Bose-Chaudhuri-Hochquenghem, and BCH (MLC). ECC is important in handling program and read disturb situations, however, ECC algorithm may not be able to prevent failure if too many bits have errors.

Newer, higher density flash memory exhibits even more volatility (including small lithography SLC), requiring more sophisticated algorithms and higher bit ECC. For example, the latest introduction of Freescale's i.MX 7 supports 62 bits of BCH ECC for raw NAND.

Attempting to perform multi-bit error detection and correction in host software can be problematic, extremely slow, and is best performed by dedicated hardware. [3, 16]

Resident or Removable?

Flash memory is either resident (soldered onto the board) or removable.

Resident flash is packaged on a board, much like the CPU and DRAM, often requiring a custom interface (both hardware and software). An example of this is resident eMMC, which is often used as the primary disk system on a smart phone. Since the media does not need to be removed, the designer has some leeway to employ

unique hardware interfaces and software-driven layouts, even though adhering to industry standards is probably in their best interest in the long run.

Removable flash, on the other hand, is often used where there is a need to exchange information with other systems using a standardized package, and therefore it must adhere to industry-accepted electrical, interface, and software file system standards. An example is the use of a removable CF (or SD) card to store photos with a FAT file system, which will later be transferred to a printer.

Connecting Flash Memory With a Host

Flash memory may appear to be a basic storage technology to manage, but what goes on under the hood is a marvel of collaborative technology; a concert of hardware and software work together to hide the complexities and subtle differences between different flash devices.

The host CPU on the target embedded board provides the brains behind the interaction of all the components on the embedded board, such as SRAM, DRAM, flash memory, I/O ports, and other devices connected to the embedded board, like the screen or keyboard. It wasn't long ago that embedded CPUs processed only 8-bit or 16-bit instructions. The rise of ARM processors for embedded platforms provides tremendous power with 32-bit and now 64-bit instruction sets.

This growing system complexity extends to more designs incorporating multiple cores and even multiple CPUs, each designated for a specific function. A perfect example is a cell phone using a dedicated applications processor to run the RTOS drivers, and applications, while a wireless connectivity processor is focused linking to various networks.

Applications use the services available in the RTOS that takes file system requests and passes them to the flash driver. This driver translates these requests to a block level interface to commands that a flash controller understands in the flash memory subsystem. This hierarchy is collectively known as the file management software stack that runs on the host (shown above the SW line in figure 9).

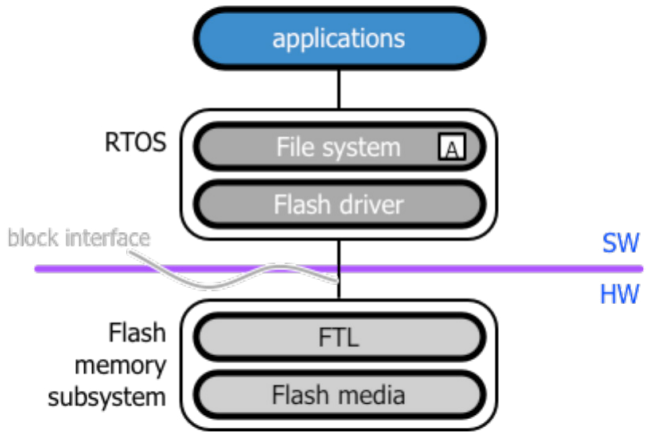


Figure 9: The file management software stack using a block interface

There are other software stacks within the system that manage various functions including network stacks, USB stacks, and so on. Through standard programmatic interfaces, each layer communicates with the next layer—in effect, allowing allows one manufacturer's code to replace those from another.

Let's look at the hierarchy of flash management from the top down. The host software components are composed of two basic parts: the operating system and associated support services and applications. Applications are the components most visible to the user, whereas the RTOS is the control program that ensures that tasks work in an organized fashion. Among other functions, the RTOS makes sure that software and hardware work well together.

A more sophisticated RTOS may provide a virtual file system (VFS) that enables multiple file systems to coexist on one platform. When an application requests a file operation (e.g., open file, read record, write record, and close file), the request would be handed off to the VFS, which commands the appropriate file system to fulfill the request.

Even with the absence of a VFS, a file system is ultimately responsible for delivering the actual request to the flash driver in the form of block read/write requests and other commands. In fact, the file system is usually not aware of the storage media underlying characteristics (be it HDD, removable flash, or resident flash) so it relies on a flash driver software component to deliver the actual request to the hardware in the form of commands.

But, where is the information actually placed on the media? A key function of a file system is to manage the placement (otherwise known as allocation) of user data and metadata shown as the letter A in the figure. How information is physically allocated on the media has direct impact to file system performance and even flash wear.

It is in the flash driver where the personalization, configuration, and optimization occurs, allowing the system to utilize the flash memory in the most efficient manner. In fact, the more the flash driver is programmed to understand the characteristics of the flash, the more it can leverage its specific features like security, discarding unused blocks, logical to physical block allocation, wear leveling, bit error detection and correction), and special techniques that optimize performance (i.e., dual-plane).

Flash memory is used as the subsystem for persistent storage. Flash must be organized to read or write in blocks and transfer data serially one bit (or multiple bits) at a time. The data is transmitted along a data path between the host and the flash by a flash controller either programmatically or via direct-memory access (DMA). [24]

The controller provides the instructions needed to perform basic reads, writes, and erases, and as more features are moved down the stack to work inside the flash. Controller feature sets are becoming more sophisticated, but still to this day, the amount of resident logic tends to be relatively simple due to limited memory on the controller chip.

Table 6 highlights what each party is responsible for.

Host software	Flash memory subsystem
Metadata	Sector numbers
Associated data	Block no longer needed (trim)
Hot and cold data	Read, write, and erase operations
Random and sequential file access	Flash the cache
Read and write block requests	ECC reporting
Application access patterns	

Table 6: Host software and flash memory subsystem responsibilities (in no particular order)

Because the host and flash subsystems are basically independent of each other, a wall exists between the host's file management software stack and the flash memory subsystem.

The Role of Standards Organizations to Make Your Decision Easier

If you are concerned about interoperability of flash devices between competitors, you're certainly not alone. Flash memory providers know that everybody wins when standards are enforced, and most of them focus on providing compatibility around the following attributes:

Functional interoperability where flash parts can be identified, connected, and responsive to standard commands from a variety of different hardware and software vendors.

Form factor interoperability where packaged flash memory (like CF or SD) can be used interchangeably. If vendors don't cooperate or support the standard, a customer would have difficulty finding memory cards that fit right or be able to transfer data contents between systems.

Some notable standards bodies include:

Open NAND Flash Interface working group (ONFI): *These standards are used to define NAND standards that encompass physical interface, commands for reading, writing, and erasing flash, and rules for specific flash part identification. Currently, ONFI has over 100 members, and the latest specification is revision 3. [25]*

SD Association: *This organization of vendors is focused on SD memory card standards and SDE host devices manufactured by over 400 brands worldwide. With the latest high-capacity SD drives, the SDXC specification supporting removable devices up to 2 TB and the UHS-II bus interface enabling speeds up to 312 MB/sec. [26]*

JEDEC standards committee: *JEDEC, consisting of over 300 companies, has been around for more than 50 years, providing standards for a wide assortment of technology. Specifically for flash memory, JEDEC has recently been focused on interoperability standards for SSDs, eMMC, and UFS. [10]*

Even with these standards in place, flash vendors offer an astonishing array of different features, quality, pricing, availability, capacity, and performance.

So, What Flash Should You Use?

Finding the ideal flash technology for your next mobile or embedded project depends, obviously, on the needs of your end user. In this paper we have highlighted a number of key differences between the basic subgroups of flash and pointed out some of the problems and pitfalls that you need to be aware of when you're making a choice. Ultimately, the unique combination of performance, form factor, endurance, reliability, and features and cost offered by a particular vendor will determine which part is the right one for your device.

Hopefully this paper has helped provide a framework for making this important decision. Following these steps should point you in the right direction:

5. Clearly identify and prioritize the customer's usage patterns.
6. Use those patterns to drive board-level design considerations (board space, power requirements, and so on) to determine the best flash memory to use.
7. Research the right software components to support your customer's needs (you wouldn't want to choose an RTOS that doesn't support your customer's requirements to run "Angry Birds").
8. Once the system is integrated (board, components, flash memory, and software), run extensive benchmarks and other integrity validation to minimize any risk of customer returns.
9. Start planning for flash part obsolescence for your next board design.

The Next Generation: Embedded Object Storage

The flash driver passes data through a block interface in a similar way that has been in existence for many years since the HDD days. This tried-and-true method of passing data via fixed-size blocks may not be efficient, but it works.

What if the host's file system was aware of how blocks were allocated on the media by the flash controller? And what if parts of the file system code could be offloaded to run inside the controller and consequently reduce the information round trips between the host flash memory subsystem?

Figure 10 shows more of an object storage design that basically treats storage of data as objects rather than as a file hierarchy managed as blocks.

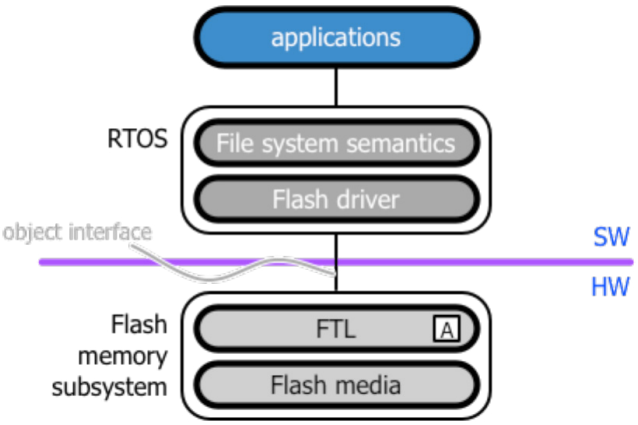


Figure 10: The file management software stack using an object interface

The basic idea that the block interface wall is replaced with a more intelligent object interface where the allocation software logic, shown with the letter A in the figure is actually pushed down to run in the FTL on the flash controller.

Table 7 shows that there's more collaboration between each party and, if implemented correctly, should require no changes to application code that requires file system services.

Host software
Flash memory subsystem
Metadata (filename, data, time, and so on)
Hints from the file system
Hot and cold data
Associated data
Application access patterns
Random and sequential file access
Management of SLC and MLC partitions

Table 7: Host software and flash memory subsystem responsibilities with object storage (in no particular order)

Flash Memory for Connected Devices

The future for reliable flash memory looks bright, especially with connected embedded devices. According to Seagate, by 2025 more than 40 billion devices are expected to be web-connected with the bulk of IP traffic driven by IoT (non-PC) devices. These embedded devices will be responsible for transmitting at least 20 trillion gigabytes of data and you can bet much of that data is expected to live in local flash memory.

Of the expected 40 billion IoT devices, the automotive sector alone is estimated to grow in excess of 3.5 billion device units in 2025, compared to just 200 million in 2014. [27] As a result, the choice of flash memory is getting more complex based on a number of factors including capacity, reliability, lifetime of the device in the field, and how long data needs to be stored on the device.

References

1. Baek, Seungjae, Jongmoo Choi, Donghee Lee, and Sam H. Noh. "Minimizing User Perceived Latency in Flash Memory File System via Uniformity Improving Paging Allocation." *ResearchGate*. April 15, 2010. http://www.researchgate.net/publication/228796748_Minimizing_User_Perceived_Latency_in_Flash_Memory_File_System_via_Uniformity_Improving_Paging_Allocation.

2 Stack Exchange. "What is the difference between Flash memory and EEPROM?" *Electrical Engineering* (Blog). May 14, 2013. http://electronics.stackexchange.com/questions/69234/what-is-the-difference-between-flash-memory-and-eeprom.

3 iSuppli. "Global shipments of HDDs and SSDs in PCs from 2012 to 2017 (in millions)." *Statista* (2015). http://www.statista.com/statistics/285474/hdds-and-ssds-in-pcs-global-shipments-2012-2017.

4. Seagate Technology. "Adaptive Memory Technology in Solid State Hybrid Drives." http://www.seagate.com/tech-insights/adaptive-memory-in-sshd-master-ti.

5. Wei, Michael, Laura Grupp, Frederick Spada, and Steven Swanson. "Reliably Erasing Data From Flash-Based Solid State Drives." *UCSD*. February 15, 2011. https://www.usenix.org/legacy/event/fast11/tech/full_papers/Wei.pdf.

6. Mellor, Chris. "TLC Flash Gets Tender Loving Care from DensBits Boffinry." *The Register*. May 2, 2012. http://www.theregister.co.uk/2012/05/02/densbits_tlc.

7. Wind River. "ARINC 653: An Avionics Standard for Safe, Partitioned Systems." *IEEE-CS Seminar*. June 4, 2008. http://www.computersociety.it/wp-content/uploads/2008/08/ieee-cc-arinc653_final.pdf.

8. Simmons, C. "Reducing JFFS2 Mount Time." *2Net*. May 24, 2009. http://www.2net.co.uk/tutorial/jffs2-summary.

9. Toshiba. "NAND vs. NOR Flash Memory: Technology Overview." *Toshiba America*. May 21, 2011. http://www.mitchellcreativegroup.com/samples_misc/toshibaflash/techoverview.pdf.

10. Panasonic, Samsung, Sony, Toshiba. "Introducing Next Generation Secure Memory Technology." March 2012. http://us.seeqvault.com/docs/NSM_White_Paper_20121116r2.pdf.

11. Athow, Desire. "SeeQVault: Next Generation Secure Memory Initiative Gets Real." *ITPro Portal*. July 9, 2013. http://www.itproportal.com/2013/09/07/seeqvault-next-generation-secure-memory-initiative-gets-real.

12. Oshita, Jyunichi. "TDK's Industrial SSD Contains Control Chip." *Nikkei Technology*. March 22,2012. http://techon.nikkeibp.co.jp/english/NEWS_EN/20120322/209836.

13. Nelson, Scott. Toshiba. "MLC and SLC NAND flash design tradeoffs. *Electronic Products*. September 8, 2008. http://www2.electronicproducts.com/PrintArticle.aspx?ArticleURL=FAJH_Toshiba_Sep2008.html.

14. Yam. Marcus. "Intel, Micron First to Triple Level Cell 25nm NAND." *Tom's Hardware*. August 18, 2010. http://www.tomshardware.com/news/nand-flash-ssd-tlc-3bpc,11102.html.

15. Inoue, Atsushi, and Doug Wong. "NAND Flash Applications Design Guide." *Toshiba America*. April 2003. http://139.138.48.19/pdf/NAND/Toshiba/NandDesignGuide.pdf.pdf.

16. Jurenka, Mark G. "Exploring Managed NAND Media Endurance." *Boise State University Graduate College*. April 6, 2010. http://scholarworks.boisestate.edu/cgi/viewcontent.cgi?article=1090&context=td.

17. IHS Technology. "A Tale of Two Memories: NAND eMMC Hits a Record Year, While NOR Flash Shrinks Further." *IHS Technology*. February 24, 2014. http://press.ihs.com/press-release/design-supply-chain/tale-two-memories-nand-emmc-hits-record-year-while-nor-flash-shrin.

18. JEDEC. "Embedded Multi-Media Card (eMMC), Electrical Standard (5.1)." *JEDEC*. February 2015. http://www.jedec.org/standards-documents/docs/jesd84-b51.

19. IHS Technology. "eMMC NAND Flash Memory Remains Viable Despite UFS Entry." *IHS Technology*. March 28, 2012. https://technology.ihs.com/395032/emmc-nand-flash-memory-remains-viable-despite-ufs-entry.

20. JEDEC. "Universal Flash Storage (UFS)." *JEDEC*. 2013. http://www.jedec.org/standards-documents/focus/flash/universal-flash-storage-ufs.

21. Armasu, Lucian. "Samsung, Xiaomi Moving To Next-Generation UFS 2.0 Memory Standard In 2015." *Tom's Hardware*. November 14, 2014. http://www.tomshardware.com/news/samsung-xiaomi-ufs-memory-standard,28076.html.

22. Micron. "Small-Block vs. Large-Block NAND Flash Devices. TN-29-07." *Micron*. http://www.micron.com/~/media/Documents/Products/Technical%20Note/NAND%20Flash/tn2907.pdf.

23. Wilcox, BD. "SSD Data Retention for Archived Drives." *Intel support community*. June 8, 2010. https://communities.intel.com/thread/13783.

24. Brown, Charles. Intel. "Dual-Plane Flash Memory Allow Wireless Systems to Support 3G Services." *Wireless Design Online*. September

24, 1999. http://www.wirelessdesignonline.com/article.mvc/Dual-Plane-Flash-Memory-Allow-Wireless-System-0001.

25. ONFI. "ONFI Flash Interface Specification. Revision 4.0." *Open NAND Flash Interface*. April 2, 2014. http://www.onfi.org/specifications.

26. Wikipedia. "Secure Digital." *Wikipedia*. http://en.wikipedia.org/wiki/Secure_Digital.

27. Yu, Ellen. "IoT to Generate Massive Data, Help Grow Biz Revenue." *ZDNET*. July 28, 2015. http://www.zdnet.com/article/iot-to-generate-massive-data-help-grow-biz-revenue

About the Author

Ken Whitaker, *Vice President, Engineering*

Ken has over 25 years of software development leadership experience in a variety of technology roles and industries ranging from shrink wrap, enterprise, to embedded markets. At Datalight, Ken leads the engineering, QA, technical support, and project management functions. He has degrees in Mathematics and Fine Arts from James Madison University and advanced Computer Science coursework towards a Masters degree from Virginia Polytechnic University (Virginia Tech).

Ken is an active PMI® member, Project Management Professional (PMP)® certified, and a Certified ScrumMaster (CSM). Ken is a frequent presenter at industry events and has authored two books: "Managing Software Maniacs" and "Principles of Software Development Leadership."

About Datalight

Datalight products have delivered proven reliability in hundreds of millions of devices in demanding product categories like automotive, medical, retail, industrial automation and military/aerospace. When data integrity, device lifespan, and design flexibility matter, the world's leading device manufacturers invest in solutions from Datalight. Our product line includes:

- **Reliance Edge™**–a tiny, power failsafe file system for Internet of Things devices
- **Reliance Nitro™**–a high-performance, power failsafe file system
- **FlashFXe™**–software acceleration for managed flash
- **FlashFX® Tera**–comprehensive software management for raw flash.

LOGIC TECHNOLOGY

J. F. Kennedylaan 18
5981 XC Panningen
The Netherlands

Tel +31 77 307 8438
Fax +31 77 307 8439

www.logic.nl
info@logic.nl

Bunsenstrasse 18
81735 Munich
Germany

Tel +49 89 1436 7945
Fax +49 89 6379 9752

www.LogicTechnology.de
info@LogicTechnology.de