1

Mass Storage Basics

A mass-storage device is electronic hardware that stores information and supports a protocol for sending and retrieving the information over a hardware interface. The information can be anything that can be stored electronically: executable programs, source code, documents, images, spreadsheet numbers, database entries, data logger output, configuration data, or other text or numeric data. Mass-storage devices typically store information in files. A file system defines how the files are organized in the storage media.

In Windows computers, mass-storage devices appear as drives in My Computer. From Windows Explorer, users can copy, move, and delete files in the devices. Program code can access files using file-system APIs or .NET's File class.

When to Use a Storage Device

Implementing a mass-storage function is a solution for systems that need to read or write moderate to large amounts of data.

If the device has a Universal Serial Bus (USB) interface, any PC or other USB host can access the storage media. Generic USB mass-storage devices include the hard drives, flash drives, CD drives, and DVD drives available from any computer-hardware store. Table 1-1 lists popular device types. These devices have just one function: to provide storage space for the systems they connect to.

Another type of USB mass-storage device (or storage device for short) is the special-purpose device with storage capabilities. For example, a camera can capture images and store the images in files. A data logger can collect and store sensor readings in files. A robotic device can receive files containing configuration parameters. With the addition of a USB mass-storage interface, any of these devices can use USB to exchange files with PCs and other USB hosts.

Generic storage devices are readily available and inexpensive. Unless you're employed by a storage-device manufacturer, there isn't much point in designing and programming your own generic devices. But special-purpose USB storage devices are useful in many embedded systems, including one-of-a-kind projects and products manufactured in small quantities.

Another option for some systems is to add USB host-controller hardware and mass-storage firmware. The embedded system can then store and read files in off-the-shelf USB storage devices.

Benefits

Adding storage-device capabilities to a system has several benefits:

- With a USB device controller, a system can make the contents of its storage media available to any PC or other USB host computer.
- File systems provide a standard way to store and access data. A PC or other USB host can format the media in a USB storage device to use the FAT16 or FAT32 file system. When the device is connected to a PC, the operating system enables reading and writing to files. Users can access the files without having to install and learn a vendor-specific application.
- Storage media is readily available. Flash-memory cards are convenient and have enough capacity for many applications. Some cards require only a few port pins to access. Devices that need large amounts of storage can interface to hard drives.

Table 1-1: Common USB mass storage devices use a variety of storage media.

Device	**Storage Media**	**Local CPU Interface to Media**	**Removable Media?**
Hard drive	Hard disk	ATA	No
CD drive	CD	ATA + ATAPI	Yes
DVD drive	DVD	ATA + ATAPI	Yes
Flash drive	Flash memory	Local CPU data bus	No
Flash-memory-card reader/writer	Flash memory	SPI, MultiMediaCard bus, SD-Card bus	Yes

Other Considerations

A storage device isn't the solution for every application, however.

- Mass-storage firmware is complex. A USB mass-storage device must support the USB protocols required for all USB devices as well as class-specific mass-storage protocols. If the device firmware needs to create, read, or write to files and directories on its own (not via the USB interface), the firmware must also support a file system. For some applications, a different USB class or a vendor-specific protocol would require less time and expense to implement.
- USB mass-storage devices transfer data using bulk transfers. These provide the fastest transfers on an otherwise idle bus but have no guaranteed timing or bus bandwidth. If your device needs precise timing in transferring data, the mass-storage class isn't appropriate.
- A storage device should have one mass-storage master at a time. The master, or mass-storage host, is the computer that reads and writes to the storage media. Special-purpose mass-storage devices can function as masters on their own and can also permit a PC or other USB host to function as the master. If one master adds, deletes, or changes a file and the other master isn't aware of the changes, confusion or worse problems can result. Devices that support two masters can have a manual or electronic switch to enable one master at a time, or a device can use firmware protocols to inform the host when the media's contents have changed. For some designs, another approach without this added complexity makes more sense.

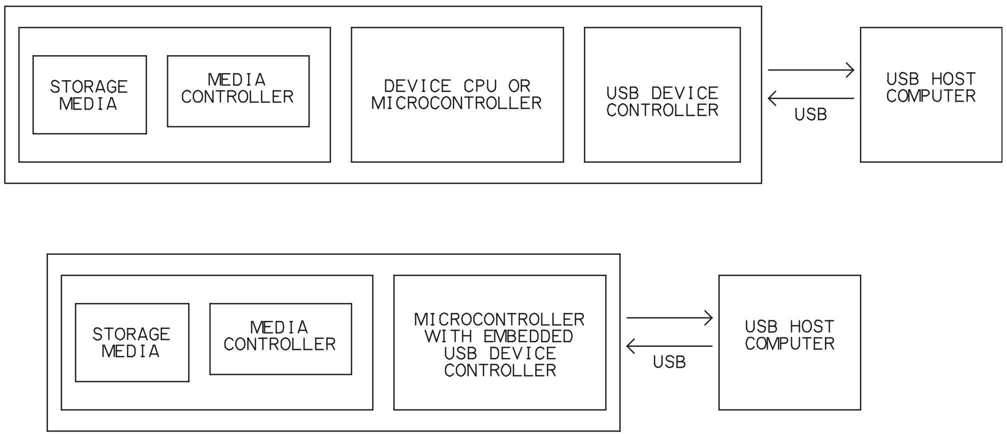

Figure 1-1: A USB mass-storage device contains storage media, a media controller, a device CPU or microcontroller, and a USB device controller, which can be on a separate chip or embedded in a microcontroller.

Alternate approaches for USB devices that transfer generic or vendor-specific data include the human-interface device class, a device accessed via a virtual COM port, or a generic or vendor-specific driver.

Requirements

Adding storage capabilities and a USB interface to an embedded system requires hardware and firmware to support accessing the storage media and communicating over the USB interface.

Devices

An embedded system that functions as a USB mass-storage device requires the following hardware (Figure 1-1):

- A microcontroller or other CPU or intelligent hardware to manage the embedded system's operation.
- A USB device controller, which can be embedded in a microcontroller chip or on a separate chip that interfaces to a CPU or microcontroller.
- A generic hard drive, flash drive, or other media that interfaces to the device's CPU.

In a USB mass-storage device, the hardware or firmware must perform the following functions:

- Detect and respond to generic USB requests and other events on the bus.
- Detect and respond to USB mass-storage requests for information or actions from the device.
- Detect and respond to SCSI commands received in USB transfers. These industry-standard commands read and write blocks of data in the storage media, request status information, and control device operation.

In addition, devices that create, read, or write to files and directories on their own (not via a USB host) must implement a file system. A file is a named collection of data. A directory structure provides an index to the files. Popular file systems for embedded systems include FAT16 and FAT32.

Two popular types of storage media for embedded systems are flash-memory cards and hard drives. A flash-memory card contains flash-memory chips to provide storage, a controller that manages reading and writing to the memory, and an interface to the outside world. Common types of flash-memory cards includes the MultimediaCard (MMC), Secure Digital (SD) Card, and CompactFlash® (CF®) card. A hard drive contains a hard disk that provides storage, drive components to perform functions such as spinning the disk and positioning the heads, a drive controller, and an interface to the outside world. An embedded system that accesses flash-memory cards or hard drives must have a microcontroller or other CPU or intelligent hardware to manage communications with the cards or drives.

This book focuses on block storage devices, where data is transferred in blocks of defined sizes. USB hard drives and flash drives are block storage devices. Other devices are stream devices, where each data transfer is a sequence, or stream, of data that can be any length. An example of a stream device is a modem that carries voice communications.

Embedded Hosts

An embedded system that functions as a USB host for flash or hard drives requires the following hardware (Figure 1-2):

- A microcontroller or other CPU or intelligent hardware to manage the embedded system's operation.

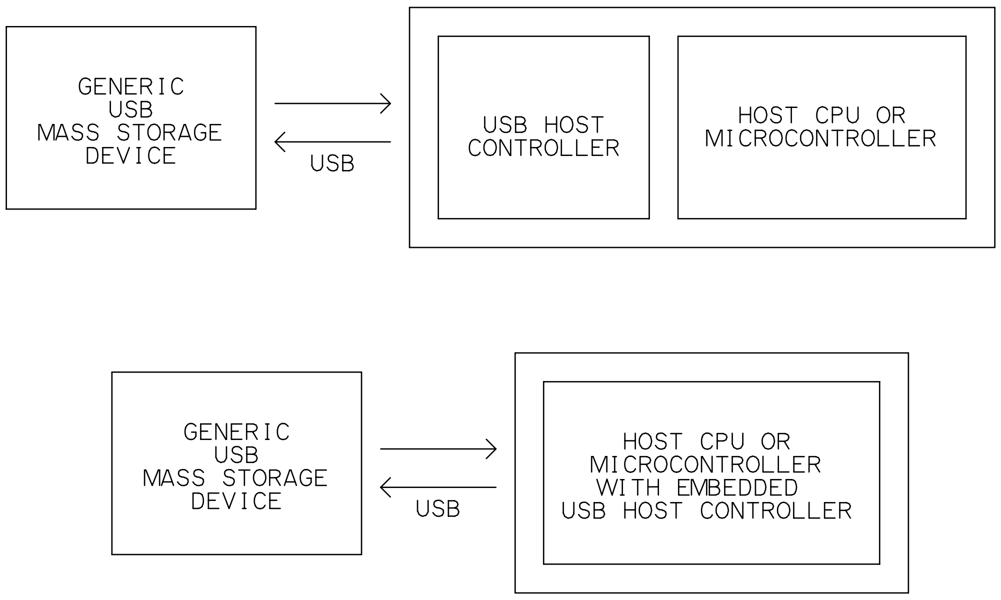
GENERIC USB MASS STORAGE DEVICE
USB
USB HOST CONTROLLER
HOST CPU OR MICROCONTROLLER
GENERIC USB MASS STORAGE DEVICE
USB
HOST CPU OR MICROCONTROLLER WITH EMBEDDED USB HOST CONTROLLER

Figure 1-2: To access generic USB mass-storage devices, an embedded system must contain a USB host controller, which can be on a separate chip or embedded in a microcontroller.

- A USB host controller, which can be embedded in a microcontroller chip or on a separate chip that interfaces to the CPU, microcontroller, or other intelligent hardware.
- A generic hard drive, flash drive, or other media connected to a USB port on the host.

The hardware or firmware in an embedded USB mass-storage host must provide the following functions:

- Issue USB requests and initiate other events on the bus to identify attached devices and manage traffic and power on the bus.
- Issue USB mass-storage requests that ask for status information or specify actions for the device to perform.
- Issue SCSI commands in USB transfers. The commands read and write blocks of data in the storage media, request status information, and control the device operation.
- Support a file system to access files in the media.

Figure 1-3: USB flash drives provide convenient storage that PCs and other USB hosts can access.

Selecting a Media Type

The storage media is the physical entity that holds a device's data. In embedded systems, a storage device's media is typically separate from the system's program memory, which stores the code executed by the system's CPU. Over time, various storage technologies and form factors have come and gone in popularity. Currently popular technologies include hard drives, CD/DVD drives, flash-memory cards, and USB flash drives (Figure 1-3). Other names for a USB flash drive (UFD) include USB key, pen drive, ThumbDrive®, DiskOnKey®, and JumpDrive®.

The different media types vary in the hardware and circuits required to access the media, the ability to erase and rewrite, methods of write protection, whether the media is removable from its drive, and interface options for external CPUs.

For many devices, flash memory is a good choice for storage media. Flash-memory cards are physically small, can store moderate amounts of data, and manage the low-level protocols for accessing the memory. Some cards require only a few port pins to access. With the addition of a USB device controller and supporting firmware, USB hosts can access the data in a device's flash-memory card. Users can also remove a card from the device and insert the card in a card reader attached to a PC or other computer. Flash memory consumes less power than other media types. When attached to a USB host or hub, a typical flash-memory storage device can receive all of its power from the bus.

Hard drives are the cheapest per byte and can hold massive quantities of data. CD and DVD drives are less common in embedded systems because embedded applications tend to require media that is easily erased and rewritten. CD-RW, DVD-RW, and DVD+RW discs can be erased and rewritten, but not as easily as magnetic media.

A device that contains a USB host controller and supporting firmware can access ordinary USB flash drives and hard drives. Because a USB host must manage the bus, USB host programming is more complex than USB device programming. But for some applications, the ability to store data in generic drives makes the increased complexity worthwhile.

Drive Mechanisms

Hard disks require a drive mechanism to spin the disks and position the read and write heads (Figure 1-4). A hard drive contains a stack of platters. Each platter has magnetic storage media arranged in concentric circles, called tracks, on both sides. Each surface of a platter has a head positioned above the platter's surface. The head can read or write to the bit of data directly opposite the head.

An area on a drive can be identified by cylinder, head, and sector. A cylinder is a stack of tracks of the same diameter. Each surface has a head, so the head identifies a surface on a platter. A sector is a portion of a track and contains the smallest addressable quantity of data in the media. All sectors in a drive have the same capacity, typically 512 bytes.

The drive mechanism spins the disks and moves the heads to requested tracks. When a requested sector on a spinning disk passes under the head,

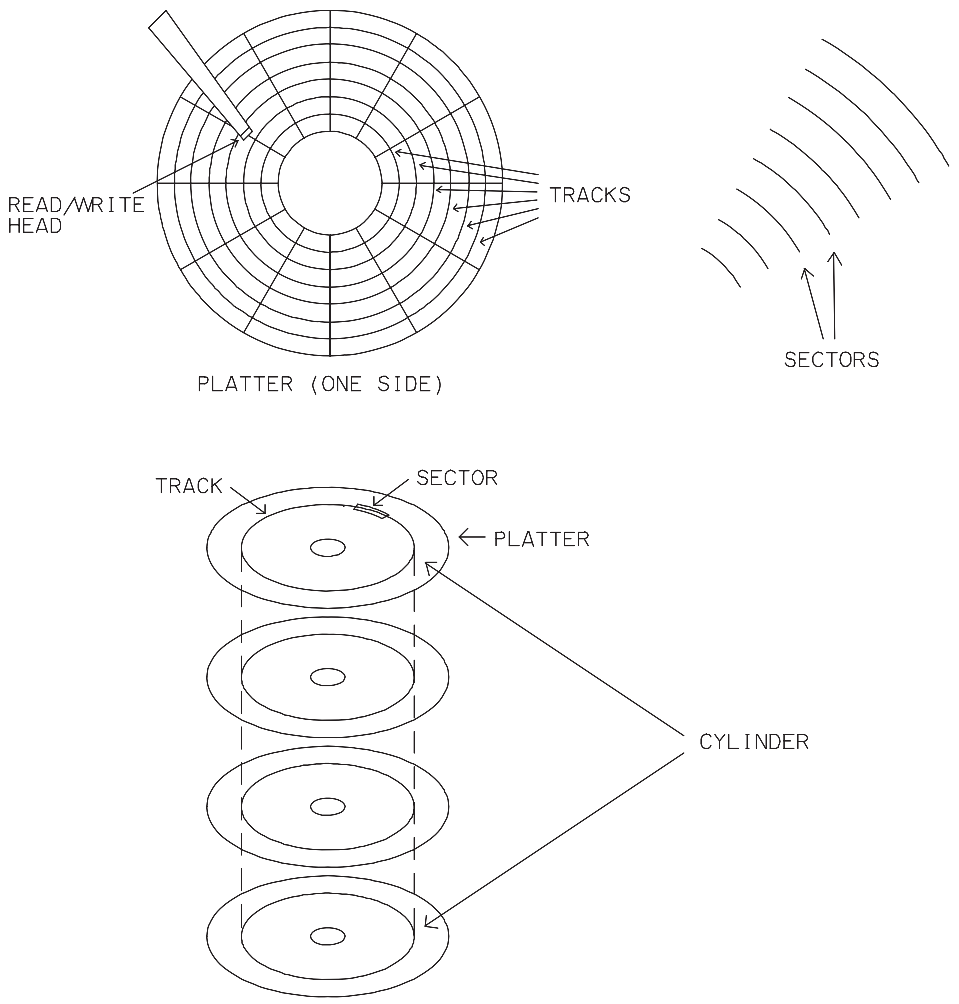

Figure 1-4: A hard drive contains multiple platters. Each side on a platter has circular tracks containing magnetic media and a read/write head. A cylinder consists of all of the tracks with the same diameter on all of the platters.

the head performs the read or write operation on the media. The head reads and writes a minimum of a sector's data in each read or write operation.

Flash memory resides in chips. Accessing flash memory requires no moving parts. USB storage devices with flash memory don't have mechanical drives, but the term *flash drive* for these devices has stuck.

Addressing Methods

All USB drives and other drives of recent vintage support logical block addressing (LBA). With LBA, blocks of storage capability are numbered sequentially beginning at zero. All blocks have the same size, again typically 512 bytes. The logical block address is often referred to as a sector address because the block size equals the capacity of a sector in a hard drive. To access the media, software specifies the logical block address to read or write to. For hard drives, the drive's controller translates each LBA to a cylinder, head, and sector on the drive. For flash drives, the drive's controller translates each LBA to a block, page, and column in the memory array. The sequence of logical block addresses doesn't have to correspond to the physical locations of the sectors in a drive or the memory in a chip. All that matters is that the media's controller knows what area of storage corresponds to each address.

In older systems, software accessed storage media using CHS addressing, where the software specifies a cylinder, head, and sector number to read or write to. A storage device can support both CHS addressing and LBA.

Compared to CHS addressing, LBA is simpler, more flexible, and supports larger capacities. File-system drivers in embedded systems are unlikely to need to use CHS addressing.

Reading and Writing Considerations

Storage media varies in the available methods of write-protecting the contents, support for erasing, and copy-protection technologies.

Write Protection

The storage media, drive mechanism, circuits, or a manual switch can permit or forbid writing to the media. For example, a flash-memory controller can forbid writing to all or a portion of the memory. Or a manual switch on a flash-memory card can inform the host that the media shouldn't be erased or overwritten. Higher-level software in the mass-storage master can also control access to data on a storage device.

Erasing

The media in a hard drive can be erased and rewritten virtually endlessly, while flash memory can survive 10,000 or more erase cycles, depending on the technology. Some memory cards contain programmed ROM chips, which can't be erased and rewritten.

The flash memory used in storage devices must be read and written in pages and erased in blocks. The page size for read and programming (write) operations is typically either 528 bytes (small block) or 2112 bytes (large block). A 528-byte page can hold one 512-byte sector and 16 additional bytes for error-correcting codes (ECC), address-mapping information for use in wear leveling, and other information. A 2112-byte page holds four 512-byte sectors with 16 additional bytes per sector. Newer memory chips tend to use large blocks.

The block size for erase operations is much larger than the page size for reading and writing. In the past, blocks of 16 KB and 32 KB were common, while current flash memory has erase blocks of 128 KB or 256 KB. Before writing to previously programmed memory, the area to be written must be erased. So to write even a single byte to a previously programmed area, the memory controller must erase an entire erase block and then program a page's contents back into the memory.

The controllers in flash-memory cards manage the erase operations and enable device firmware to work with 512-byte blocks. To write a byte to a flash-memory card, device firmware typically reads 512 bytes into a buffer, changes the byte to be written, and writes the buffer back to the memory card. The card's controller handles the erase and write operations.

The controllers in flash-memory cards use wear-leveling techniques that help extend the useful life of the memory array by spreading erase/write cycles evenly among all of the memory cells. A file-system driver that accesses raw flash-memory chips can implement wear leveling as well.

Security

Some media types have built-in copy-protection capabilities. For example, an SD Card can be configured to require authentication before allowing access to the card's contents, and a card can restrict the number of allowed copies.

Removable Media and Devices

A device can have removable media, and an entire device can be removable from the computer that communicates with the device.

Removable Media

In a drive with removable media, users can easily insert and remove media in the drive. CD and DVD drives have removable media because you can easily swap discs. A memory-card reader with a card slot has removable media. Hard drives and flash drives have non-removable media because you can't easily remove the hard disk from its drive or the flash memory from its circuit board. A device reports whether it has removable media in the response to a SCSI INQUIRY command. Some flash drives with non-removable media report that they have removable media. Chapter 6 has more about the INQUIRY command.

Removable Devices

An entire storage device can also be removable or non-removable from the computer that accesses the drive. USB drives are removable. An internal drive is considered non-removable because removing the drive requires more work than detaching a cable.

Managing Removal

A user can detach a USB device or remove a flash-memory card at any time. If a device or card is removed while the host is writing to the media, the device and host should detect the removal and handle it as gracefully as possible.

Hardware Interfaces

A storage device can support one or more interfaces to its storage media. In most cases, the device's CPU doesn't access the media directly. Instead, the CPU communicates with an intelligent controller embedded in a drive or flash-memory card. In devices that support USB, the CPU also interfaces to a USB device controller.

Hard Drives

When you need a lot of storage, a hard drive is the most economical choice. At this writing, a megabyte of hard-drive storage is 50 to 100 times cheaper than a megabyte in a flash-memory card. Prices for both media continue to fall, and the price differential may change over time, but for the near future, hard drives are likely to continue to be the favored solution for storing very large amounts of data.

Technology

Because they use mechanical drive components, hard drives tend to be more fragile than completely electronic media such as flash memory. Modern drives are much more rugged than in the past, however. For embedded systems that need to fit in a small space, tiny hard drives are available in the USB key-drive form factor and in Type II CompactFlash cards.

Interfaces

The most common interface between a hard drive and its CPU is the Parallel AT Attachment (ATA) interface, also known as the Integrated Drive Electronics (IDE) interface. A drive that uses ATA must have an intelligent controller embedded in the unit. The ATA specification defines the cables and connectors, signals, and registers and commands for communicating with the drive's controller. ATA devices must support logical block addressing. A single ATA interface on a host computer can connect to up to two storage devices. The host computer communicates by reading and writing to registers in the device.

ATA with Packet Interface (ATAPI) is an extension to ATA that defines a protocol for sending SCSI and other commands to an ATA device in structures called command packets. CD and DVD drives use the ATAPI protocol. More information about ATA/ATAPI and links to the standard documents are at www.ncits.org.

Flash Memory

Flash memory is non-volatile, electrically erasable storage available as chips and in cards that incorporate memory chips and a controller.

Technology

Both flash memory and EEPROM provide non-volatile, electrically erasable storage. Compared to EEPROM, flash-memory cells are physically smaller, can withstand more erase/write cycles, and are cheaper to manufacture. The main disadvantage of flash memory is that unlike EEPROM, flash memory is erasable only in blocks, not by individual byte. Even so, for most storage devices, flash memory is the more practical choice, while EEPROM is useful for storing infrequently changed configuration settings.

Two flash-memory technologies in popular use are NOR and NAND.

NOR flash is suited for storing program code, where the CPU wants fast read access but rarely writes to the memory. NOR flash has fast read times but slow erase and write times. NOR flash has low density, so large amounts of storage may require multiple chips. To access NOR flash, a CPU uses the same data and address lines used to access other parallel memory chips.

Storage devices use NAND flash, which has fast erase and write times. NAND flash also has lower power consumption and is much cheaper than NOR flash. A CPU accesses NAND flash chips via data lines and command and address registers. NAND flash has high density, so large amounts of memory can fit in a small package. The advantages of NAND flash are so attractive that some devices use NAND flash for program memory along with a RAM cache to improve performance.

Three varieties of NAND flash are Old Single-level Cell (SLC), New SLC, and Multi-level Cell (MLC). MLC memory stores multiple bits in each cell and is popular because it's cheaper to manufacture. However, compared to SLC memory, MLC memory supports fewer erase cycles, has slower write times, and consumes more power. These are the typical number of erase cycles supported by each memory type: Old SLC: 1,000,000, New SLC: 100,000, MLC: 10,000.

Wear Leveling

Wear-leveling techniques can extend the useful life of flash memory by writing to different physical locations in each erase/write cycle. A typical write operation accesses only a portion of the memory in a flash-memory card. Writing to different locations with each write operation helps to spread the erase/write operations evenly to all areas of the memory and extends the life of the memory as a whole. A card that uses chips rated for 10,000 erase cycles can thus withstand a much greater number of erase operations if each operation erases a portion of the memory. Wear leveling is especially important if the chip stores files or structures such as file allocation tables (FATs) that are rewritten frequently.

To accomplish wear leveling, firmware can map each logical address to a physical address that changes with each write operation. Higher-level firmware writes to the logical addresses, and the wear-leveling firmware translates the addresses to physical locations in the memory. For example, write operations can be mapped to the physical addresses in sequence, starting over at the beginning after reaching the end.

Error Correcting

Error correcting code (ECC) bytes enable a controller to verify data in read operations. The controller generates and stores an ECC when writing to a block and can use the code to verify the data after reading the block.

Manufacturers of flash-memory cards may implement additional protocols to help ensure data reliability. For example, under margin conditions, the controller in a Sandisk MultiMediaCard reads data back after writing to verify the write operation. If a bit is bad, the controller replaces the bit with a spare bit. The controller can also replace an entire bad block with another block. Higher-level protocols can also support error correcting via checksums sent with data.

Options for Flash Memory

Using flash-memory cards rather than raw flash chips has two advantages. The controller in the card greatly simplifies accessing the memory. And cards are easily removable and replaceable, so you can store data in multiple cards and replace cards that fail or wear out.

Things to consider when selecting a type of flash-memory card include physical size, capacities, interfacing options, data-transfer speed, power-supply voltage, and cost. Cost includes the price for the memory cards and connectors as well as any charges for specification documents, licensing, and royalties. If protecting the media's contents from writing or copying is important, some cards have this capability built in.

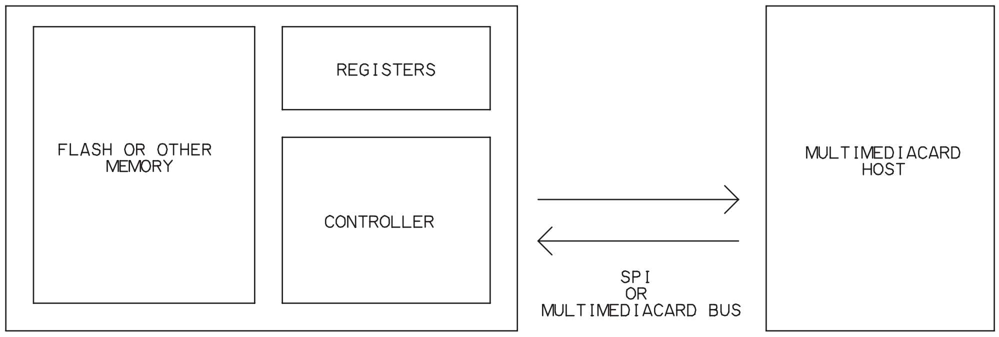

Figure 1-5: Each MultiMediaCard contains memory, registers, and an intelligent controller.

For many embedded systems, a MultiMediaCard host is a good choice because the cards are small, many microcontrollers can interface to them, and MultiMediaCard hosts have no licensing fees. A MultiMediaCard host can also communicate with SD Cards if the connector accepts the slightly thicker SD cards. Other options are an SD-Card host or CompactFlash host.

MultiMediaCard

A MultiMediaCard contains these elements (Figure 1-5):

- Memory for data storage. The memory is typically flash memory but ROM-based MultiMediaCards are also available.
- Five registers that can store configuration and status information such as valid power-supply voltages and whether the card has completed its power-up procedure.
- An interface that supports communicating via the MultiMediaCard bus and SPI.
- A controller that executes MultiMediaCard commands.

There are three classes of MultiMediaCards. The Read/Write (RW) class encompasses cards that can read and write to storage media, typically flash memory. Read-only Memory (ROM) cards support reading but not writing to the storage media. I/O cards perform additional functions beyond data storage.

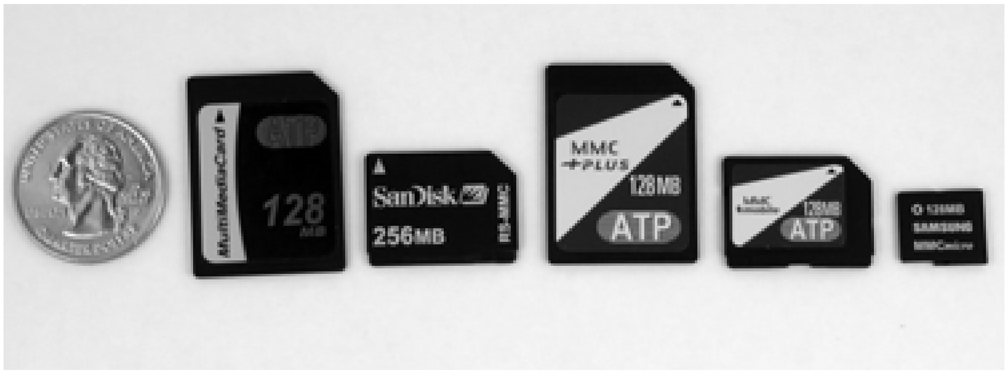

Figure 1-6: From left to right: MultiMediaCard, RS-MultiMediaCard, MMCplus, MMCmobile, and MMCmicro.

The MultiMediaCard specifications are a product of the MultiMediaCard Association (MMCA) (www.mmca.org). The MMCA board consists of over a dozen semiconductor and technology companies. The organization is dedicated to open, royalty-free standards.

The MultiMediaCard specifications are the ultimate authority on the physical interface and command set. Data sheets for specific MultiMediaCards are another helpful source of information about the interface, protocols, and card-specific information.

Packages

Table 1-2 compares the five MultiMediaCard variants. The cards are available in three form factors and with interfaces of 7, 10, and 13 pins (Figure 1-6). The original MultiMediaCard has a 7-pin interface and is about 1.25 x 1 inch in size. The RS-MultiMediaCard is functionally identical and about half the size. The MMCplus™ and MMCmobile™ add 13-pin interfaces in both form factors. The MMCmicro™ has 10 pins and is about half an inch square.

Interfacing

A MultiMediaCard can use either of two synchronous serial interfaces: the MultiMediaCard bus or SPI. Just about any microcontroller can implement either bus. The MMCplus, MMCmobile, and MMCmicro can also use a

Table 1-2: MultiMediaCards are available in several formats.

Card	MultiMediaCard	RS-MultiMediaCard	MMCplus	MMCmobile	MMCmicro
Sponsor	mmca.org				
Physical Size (mm)	32 x 24 x 1.4	18 x 24 x 1.4	32 x 24 x 1.4	18 x 24x 1.4	14 x 12 x 1.1
Pins	7		13		10
Interface	MultiMediaCard bus (serial), SPI		MultiMediaCard bus (serial or parallel), SPI		
Data Bus Width (bits)	1		1, 4, 8		1, 4
Maximum Data Transfer Rate (Mbits/sec.)	20		416		208
Maximum Clock Speed (Mbits/sec.)	20		52		52
Power Supply (V)	3/3.3/1.8				
Security	SecureMMC interface supports digital rights management				
Specification Cost	$500 or MMCA member @$2500/year		$1000 or MMCA member @$2500/year (includes all MultiMediaCard variants)		
Licensing Fees and Royalties	none				

4-bit parallel MultiMediaCard bus. The MMCplus and MMCmobile can use an 8-bit parallel MultiMediaCard bus.

An SPI host must have a clock output (SCLK), a data output (DataIn on the card), and a data input (DataOut on the card). The host must also control a unique chip-select output (CS) for each device the host communicates with.

A MultiMediaCard-bus host must have a clock output (CLK), a bidirectional pin for commands (CMD), and a bidirectional pin for data (DAT). The master uses commands to assign addresses and select cards, so the MultiMediaCard bus doesn't need a chip-select line for each card.

On power-up, a MultiMediaCard must be clocked at 400 kHz or less. When the initialization procedure is complete, the host can increase the clock frequency.

These are advantages to using SPI:

- Many microcontrollers include hardware support for SPI. The hardware support simplifies programming.
- All SPI signals are unidirectional so the host doesn't need to have bidirectional port pins.
- A variety of chips and modules in addition to MultiMediaCards have SPI interfaces. The options include EEPROMs, analog-to-digital converters, and other I/O functions. A microcontroller can thus use one bus to access multiple components.
- For interfaces that don't require error checking, an SPI host can instruct a card to ignore error checking. Error checking is mandatory with the MultiMediaCard bus.

These are advantages to using the MultiMediaCard bus:

- The host doesn't require a chip-select line for each card. Instead, addresses are assigned via firmware.
- The host can broadcast commands to multiple cards.
- The host can perform stream reads and writes, where the data isn't in defined blocks and the card or host transmits continuously until the host issues a STOP_TRANSMISSION command. SPI hosts can perform block reads and writes only.
- MMCplus, MMCmobile, and MMCmicro cards can use a parallel data bus for faster transfers.

A host selects the MultiMediaCard bus or SPI by controlling the CS pin on the card when sending the GO_IDLE_STATE command to the card. To use SPI, the host brings CS low while sending the command. To use the MultiMediaCard bus, CS remains high. All communications that follow use the selected bus.

The MultiMediaCard specification doesn't mandate power-consumption limits. A typical MultiMediaCard consumes 50 mA during read operations and 60 mA during write operations. Cards can support a low-power sleep mode when the card isn't being accessed.

Protocols

The MultiMediaCard specification defines a set of MultiMediaCard commands. The host uses the commands to retrieve information about a card and its status, to send control information to a card, and to read and write data in the storage media. An SPI host can use most of the MultiMediaCard commands. Chapter 4 and Chapter 5 have more about MultiMediaCard programming.

Fees

MultiMediaCard hosts have no licensing fees, but the MMCA charges to download the specifications. At this writing, the cost is $500 for version 3.1 of the specification and $1000 for version 4.1, which adds the MMCmobile and MMCplus variants. Those who join the MMCA at $2500/year get the specifications for no additional charge plus other benefits.

SD Memory Card

Secure Digital (SD) Memory Cards, or SD Cards for short, are similar in capability, size, and pinout to MultiMediaCards. An SD-Card host can communicate with both MultiMediaCards and SD Cards.

Compared to MultiMediaCards, SD Cards have these differences:

- In the original form factors, SD Cards are thicker than MultiMediaCards (2.1 mm versus 1.4 mm). Card connectors that accommodate both types are available. With adapters, you can use any form factor of either card type in a full-size SD-card connector.
- Some SD Cards have a manual write-protect switch.
- SD cards have additional registers with configuration and status information.
- SD Cards support additional commands, including a command that enables the host to specify a power-supply voltage.
- Unlike MultiMediaCards, SD Cards don't need to be clocked at 400 kHz or less until the card is initialized (but doing so causes no harm).

The SD-Card technology was developed by Matsushita Electric Industrial Co., Ltd., SanDisk Corporation, and Toshiba Corporation.

Packages

Table 1-4 compares the SD-Card variants. SD Cards are available in three form factors: original SD Card, miniSD™ Card, and microSD™ Card (Fig-

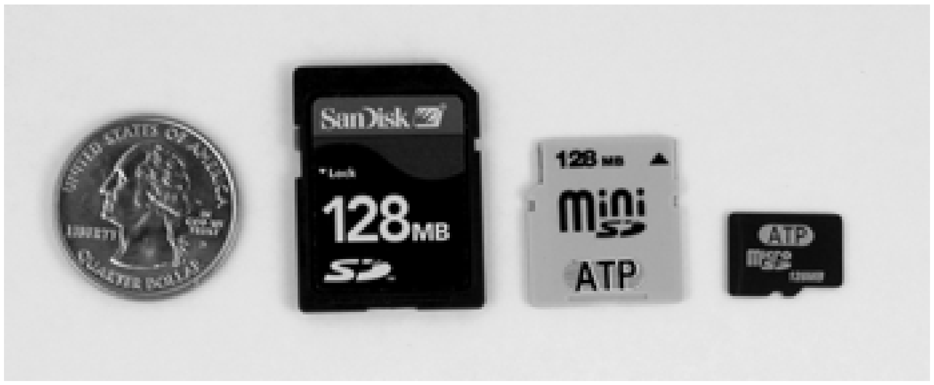

Figure 1-7: SD Cards are available in three form factors. From left to right: original SD Card, miniSD Card, and microSD Card.

ure 1-7). The form factors are similar to the options available for MultiMediaCards. A card that performs I/O functions such as modem, GPS device, or network interface is called an SDIO Card.

The optional write-protect switch is a sliding tab on the side of the card. If the tab is in the lock position, the host must write-protect the contents. The switch by itself doesn't offer protection. The firmware accessing the card is responsible for reading the state of the switch and protecting the contents when appropriate. SD-Card connectors include a pin that enables reading the switch state. The miniSD and microSD Cards don't have write-protect switches but can be inserted in an SD-Card adapter that contains a switch.

Interfacing

An SD Card can use SPI or the SD-Card bus. The SD-Card bus can use a bus width of one or four bits. The SD-Card bus can have shorter timeout values and doesn't require a clock frequency of 400 kHz or less on power up. A typical SD Card uses 65 mA to read and 75 mA to write and has a low-power sleep mode when the card isn't being accessed.

Protocols

SD Cards use the same commands and protocols defined by the MultiMediaCard specification. SD Cards also support a series of commands that are specific to SD Cards. These commands support security functions and

Table 1-3: Secure Digital (SD) Cards are available in several formats.

Card	SD Card	miniSD Card	microSD Card
Sponsor	sdcard.org		
Physical Size (mm)	32 x 24 x 2.1	20 x 21.5 x 1.4	11 x 15 x 1
Pins	9	11	8
Interface	SD Card bus, SPI		
Data Bus Width (bits)	1, 4		
Maximum Data Transfer Rate (Mb/sec.)	100 (SD Card bus); 25 (SPI)		
Maximum Clock Speed (Mbits/sec.)	25		
Power Supply (V)	2.7-3.6 or 1.6-3.6(LV)		
Security	support for digital rights management		
Write Protect Switch	optional	no	no
Specification Cost	Membership @$1000/year		
Licensing Fees and Royalties	Host/ancillary product license for $1000/year, available to members only		

enable reading additional status information and controlling a pull-up on the Card Detect pin.

Fees

Implementing an SD-Card host isn't practical for developers of inexpensive products that sell in modest quantities. Every device that contains an SD-Card host must be licensed. At this writing, it costs $1000/year to join the SD Card Association and another $1000/year for a member to license a host. Membership includes access to the SD Card specifications.

If you don't need the additional capabilities of SD Cards, a MultiMediaCard host is a less expensive option. If you use a connector wide enough to accept SD Cards, the host can communicate with both MultiMediaCards and SD Cards operated as MultiMediaCard-compatible devices.

CompactFlash

Another option for flash memory is the CompactFlash card (Figure 1-8). Like MultiMediaCards and SD Cards, CompactFlash cards contain memory, registers, and an intelligent controller. CompactFlash was introduced by SanDisk Corporation. These cards are a solution if you need to store a lot of data in a small package or need very fast transfers.

Packages

Table 1-3 compares the two CompactFlash variants. Both are 1.7 in. wide and 1.4 in. or greater in length. A Type II CompactFlash card is thick enough (about 0.2 in.) to hold a tiny hard drive. A CF+™ card is any card that has the CompactFlash form factor and contains storage media other than flash memory or performs I/O functions other than storage.

Interfacing

CompactFlash cards can use an 8- or 16-bit parallel data bus. Storage devices can use either of two modes. PCMCIA mode is based on the PC Card (PCMCIA) interface and uses an 11-bit address bus. True IDE Mode is based on the ATA-4 specification, is compatible with the IDE disk drive interface, and uses a 3-bit address bus to select registers.

A CompactFlash card can draw up to 75 mA at 3.3V or 100 mA at 5V. A CF+ card can use either of two power levels. The limits for Power Level 0 are the same as for CompactFlash, while Power Level 1 allows drawing up to 500mA at 3.3V or 5V. All currents specified are average RMS currents.

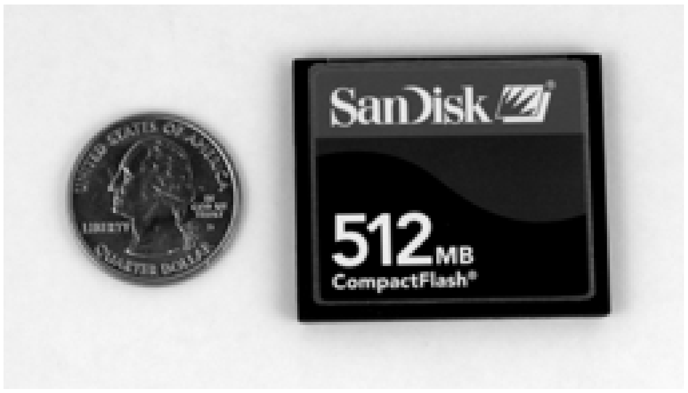

Figure 1-8: A CompactFlash card is another option for flash-memory storage.

Table 1-4: CompactFlash cards are available in several formats.

Card	CompactFlash	CompactFlash II
Sponsor	compactflash.org	
Physical Size (mm)	36.4 or greater x 42.8 x 3.3	36.4 or greater x 42.8 x 5
Pins	50	
Interface	PC Card/True IDE Mode	
Data Bus Width (bits)	8 or 16	
Maximum Data Transfer Rate (Mbits/sec.)	528	
Power Supply (V)	3.3/5	
Security	Security Mode password protection, recommended for use only in non-removable devices	
Specification Cost	none	
Licensing Fees and Royalties	Membership @$2500/year enables using the logo and trademarks	

Protocols

CompactFlash and CF+ cards with storage media are accessed much like other ATA hard drives. A series of registers store status and control information and data being transferred. The CompactFlash specification defines a CF-ATA command set for communicating with cards.

Fees

The CF+ and CompactFlash specification is available at no charge. Use of the CompactFlash logo and trademarks on products requires membership in the CompactFlash association at $2500 per year.