AN1187
APPLICATION NOTE

All you need to know before choosing a Flash Memory

CONTENTS

INTRODUCTION

Flash Memories keep the data that is saved to them even when the power is removed: they are non-volatile memories. They offer advantages over ROM and EPROM because they can be erased and reprogrammed by the microprocessor they are connected to.

The number of Flash Memories sold in 1998 is more than twice the number sold in 1995. Heavy pressure on their price since 1995 means that the cost per bit is now more competitive than ever. If you use non-volatile memories in your designs, you are increasingly likely to choose a Flash Memory.

This document gives an overview of Flash Memories and how to use them. If you are new to Flash Memories then this document should help you understand the jargon associated with Flash Memories. At the end of the document the Glossary contains descriptions of some of the terms used in this rapidly expanding industry.

This document is not going to help you choose a specific Flash Memory. Once you have read the document and understood the issues involved you will be able to go to the Product Selector and choose the correct Flash Memory for yourself.

APPLICATIONS OF FLASH MEMORIES

There are many applications of Flash Memories in existence today. Recently “MP3” took top place in the most searched for term on the world-wide web. Every MP3 digital audio-player being developed is likely to choose Flash Memory to store music. Flash Memory offers a compact, low power, rugged storage solution that CD and other discs cannot.

One of the first major markets for Flash Memories was in the PC motherboards to hold the BIOS. Many manufacturers allow users to update the BIOS by changing the Flash Memory contents. Hard Disks, CD-ROM, CD-R/W and DVD all make extensive use of Flash Memories to boot their microcontrollers and execute their internal algorithms. Networked peripherals, such as printers, ISDN routers and firewalls all use Flash Memories to allow their software protocols to be changed easily as well as storing their user parameters (IP address for example).

Mobile phones manufacturers use millions of Flash Memories every year. They benefit from being able to run software directly from their Flash Memories, reducing the amount and cost of the RAM they require. In production they are able to reprogram the Flash Memory with the latest application software at the end of the production line, rather than before the PCB is assembled. They can even assemble the PCB with test software that is later replaced.

Other consumer products, such as Set-top Boxes use Flash Memories for both software and data storage. Many Set-top Boxes are able to upgrade their software in the field. The new software is sent over the airwaves and programmed into the Flash Memory so they can remain abreast of technology advances for many years.

Engine management systems in cars could benefit from Flash Memories by allowing the engine management software to be upgraded along with a service. Economy and performance enhancements can be made even after the car has been sold.

Other embedded products, such as digital cameras and GPS receivers, can also benefit from field upgrades. Many manufacturers offer users the ability to download the latest version of software from the web and program it into their product for upgrades or performance enhancements.

Flash Memories are now the first choice for prototype and development products since the software can be quickly updated to incorporate new features. Product development using Flash Memories can be considerably faster than EPROM since the contents can be changed without dismantling the system. The cost of Flash Memories is very competitive and, therefore, the Flash Memory usually stays in the product design once it moves to production.

There are thousands of applications for Flash Memories. If you are not using a Flash Memory at the moment you may be able to use one and provide a benefit to your customers before your competitors. They are cheaper than EEPROM and more versatile than EPROM. Once you have the Flash Memory in your design, who knows what benefits you will be able you gain from it!

SYSTEM ISSUES

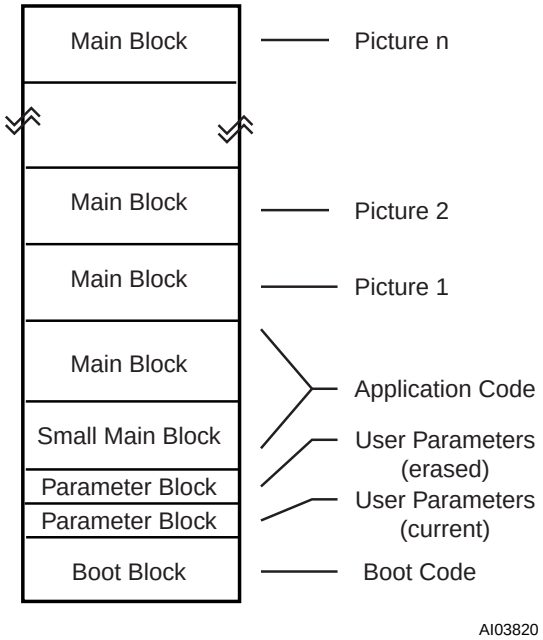
If you are going to make use of a Flash Memory in your system then you should plan how you are going to make use of the Flash Memory from the start. This will help you decide which Flash Memory you require. This section is intended to help you understand how to use a Flash Memory in your system.

Unlike RAM, you cannot write data directly to a Flash Memory, you have to program it. Furthermore, you have to program the data to a location that has been erased. And, you can only erase Blocks of data at a time. These restrictions do not severely limit the use of Flash Memory, once you have the tools in place to ensure you do not lose your data you will find them easy to use.

Most Flash Memories are divided into Blocks. Each Block can be erased separately. You can protect Blocks so that accidental program and erase operations cannot change the data. Before erasing a Block it is important that none of the data in the Block is required as all of the data will be erased. The most common Block size is 64 Kbytes, though smaller Block sizes do exist. Read and program operations work independently of the Blocks. Any byte (or word) can be read or programmed separately.

Consider Figure 1, which shows how a Flash Memory might be used in a digital camera.

Figure 1. Example of how a Flash Memory might be used in a Digital Camera



In this example the functionality of each Block has been predefined. The bottom of the Flash Memory contains the Boot Code for the microprocessor, the next two Blocks are used to store user parameters. Following this the Application Code fills two Blocks and the remaining Blocks are used to store pictures. This model is simpler than the one actually employed by most digital cameras, where the Blocks are divided into smaller partitions.

Boot Block Flash Memories are ideal for this type of application. They have different sized Blocks at one end of the address space, allowing the product to make better use of the available space in the Flash Memory. The Boot Block is usually 16 Kbytes, the Parameter Blocks are usually 8 Kbytes and the Small Main Block is usually 32 Kbytes. The Boot Block, Parameter Blocks and Small Main Block can be located at the Top or Bottom of the Flash Memory's address space. A Flash Memory with these Blocks at the top of the address space is called a "Top Boot Block Flash Memory"; if these Blocks are at the bottom of the address space then the memory is a "Bottom Boot Block Flash Memory".

Boot Block Flash Memories are usually positioned in the microprocessor's address space so that the microprocessor executes the program in the Boot Block after reset. In general Intel microprocessors boot from a memory location near the top of the address space (therefore use Top Boot Block Flash Memories) whereas Motorola microprocessors boot at the bottom of the address space (therefore use Bottom Boot Block Flash Memories). The Boot Block can be protected to prevent the Boot Code from being programmed or erased.

Parameter Blocks are useful for storing the options that the user has selected. They are small in size and erase quickly, making them ideal for rapid updating. There are usually two or more Parameter Blocks in a Flash Memory. By using two Blocks the data can be secured against inadvertent power failure. The old data in one Parameter Block does not need to be erased until the new Parameter Block has been written. If only one Block is used then there is always the risk that the user will remove the power during the erase/reprogram operation and some data will be lost.

The Small Main Block is a good place to start the Application Code. Application Code can continue into the adjacent Main Blocks. In the case of the camera, if the Application Code grows then it can occupy the space used by the first picture (so long as the picture can be moved or deleted).

The remaining Main Blocks can be used to store the user's data. Most Main Blocks are 64 Kbytes in size. It is in these Main Blocks that more optimization can take place. It is unlikely that each picture will be exactly 64 Kbytes in size. Many applications use algorithms for storing data of variable size, rather than of a fixed size. This results is better usage of the space in the Flash Memory. There are many commercially available software products to manage the data areas of the Flash Memory.

HARDWARE ISSUES

The hardware issues relevant to Flash Memories are not too dissimilar to those of EPROMs, except for the additional control line, Write Enable, that needs to be connected. The Data Sheet for Flash Memories contains all of the information needed to design a Flash Memory into a processor board. The pin connections, signal descriptions, DC and AC characteristics are described in detail. In this section some of the key parameters will be identified and an explanation of the various bus structures is given.

The first key parameter to be considered is the voltage and power requirements. Most applications of Flash Memories are battery powered and therefore power sensitive. To try and reduce the power required by electronic systems the operating voltage has fallen in recent year from 5V to 3V and now down to 1.6V. Plans for 0.9V systems in the future exist. Flash Memory is moving down this route too. Currently both 5V and 3V memories exist. Memories that have 1.6V external buses but 3V cores are also available. Parts are in development that can be programmed from a 1.6V single supply.

The amount of data storage available in a Flash Memory today ranges from about 1 Mbit (128Kb x8) to 32 Mbit (4Mb x8). These are manufactured in 0.25µm technology, shrinking this to 0.18µm is underway. Research for 128 Mbit and 256 Mbit is also taking place. The bus widths that Flash Memories are available in are 8-bits, 16-bits and, recently, 32-bits. Many of the larger bus width Flash Memories can be accessed as a smaller bus width Flash Memory as well; for example the M29F400B is an x8/x16 bus width memory, meaning it can be configured for 8-bit wide or 16-bit wide bus access.

The power requirement of a Flash Memory varies depending on the operation that is being performed. Read operations make use of Address Transition Detection; this wakes up the Flash Memory whenever it is being accessed and the address changes. The power consumption increases to about 15mA at the start of the read operation, but quickly falls to the Standby Current once the data has been read from the memory array. A Flash Memory that is being accessed at 6MHz (6 million reads per second) has a maximum average current of about 10mA; at 1MHz this will fall to about 1.6mA. During program or erase operations the current supply is higher; about 20mA is required throughout the program or erase operation.

Access time is another critical parameter for hardware designers. Currently the smaller Flash Memories (1MBit) have access times of 35ns and the larger ones (16MBit) have access times of 55ns. Low voltage (3V) Flash Memories are slower than their 5V equivalents. Memories with Burst Mode and Page Mode are available in 16MBit sizes with access times of about 100ns followed by 20ns for subsequent reads. Note that many Flash Memories have slightly slower write access times than read access times.

Flash Memories are protected from glitches corrupting their contents during power on and power off. The commands on many Flash Memories require some "Coded Cycles" to be written to the memory before program or erase commands will be accepted. The Coded Cycles are virtually impossible to issue by accident as a glitch on the address and data bus. Once the supply voltage falls below the Lockout Voltage the Program/Erase Controller will not accept any program or erase commands. In addition critical Blocks can be protected so that no program or erase operation can modify the data in these Blocks.

Flash Memories are available with Commercial, Industrial and Automotive Temperature ranges. There is a trade-off between access time and temperature range; the fastest parts may not be available over the widest temperature range (though you should still ask if you require these parts since they are often available).

Nearly all Flash Memories are only available in surface mount packages. The M29F040B is one of the only ones available in the non-surface mount PDIP package. It is not necessary to have a multi-layered PCB to use a Flash Memory; however, it is good engineering practice to have a ground plane in any design where there are signals with pulse widths less than 100ns. A decoupling capacitor should be placed near to the Flash Memory to keep V_{CC} constant during transient current requirements.

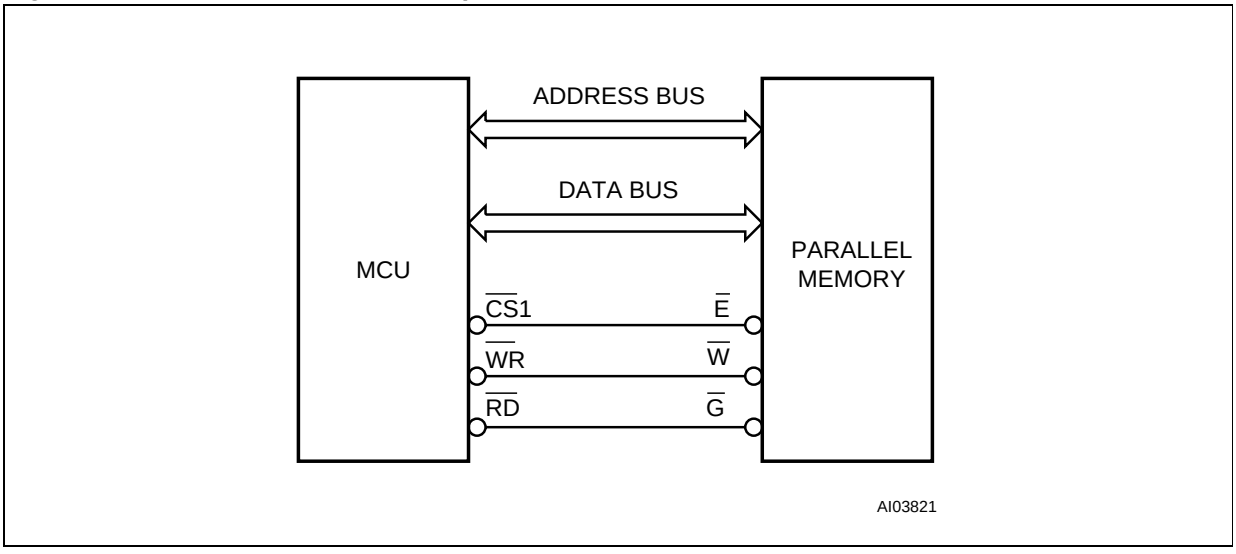
Access Mechanisms

Flash Memories come with different access mechanisms, the main three being Parallel Memories, Serial Memories and Serial NAND Memories. STMicroelectronics concentrates on manufacturing Parallel Memories, but a brief description of each is given here.

Parallel Memories are used to connect to microprocessor buses. Figure 2 gives an example of the main control lines that are required. Almost all Parallel Memories have de-multiplexed address and data buses as shown in Figure 2, although Flash Memories with multiplexed address/data buses are in development, if not available. Parallel Memories are read in three steps: firstly the address is set on the address bus; secondly the control signals select a read operation from the memory and finally the data output by the Flash Memory is read by the microprocessor. Writing to Parallel Memories is a two step process: the address and data buses are set, followed by the control signals to latch the data into the Flash Memory. Note that this is a Bus Write operation: it writes commands to the Command Interface of the Flash Memory, not to the memory array.

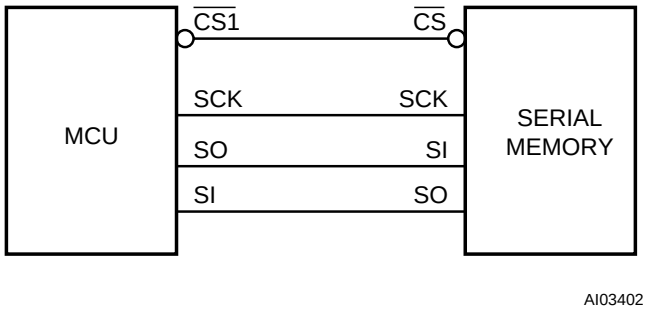
There is more behind the term “Parallel Memory” than the bus configuration. Parallel Memories are random access devices: any byte (or word) can be accessed in any order with the same access time. This may seem obvious at the moment, but read on and you will find out that Serial NAND Memories do not have this property. (Note that Burst Mode and Page Mode Flash Memories can access sequential or pages of bytes faster than normal operations, but the access time is still the same order of magnitude).

Figure 2. Connection between Microprocessors and Parallel Memories



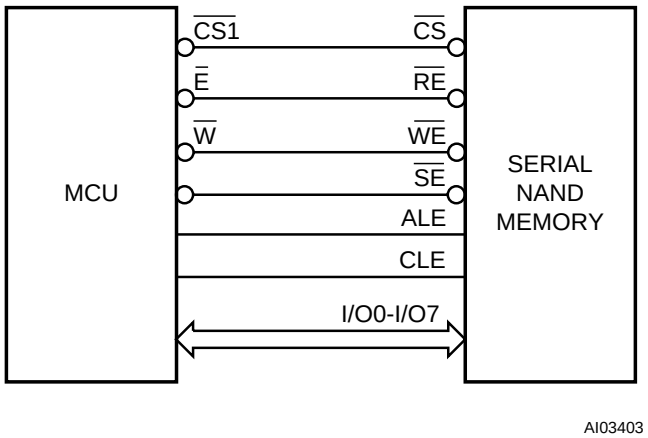
Serial Memories have a Serial Bus instead of a Parallel Bus. The SPI Bus is the most common bus used by Serial Memories. Figure 3 shows an example of a Serial Memory connected to a microcontroller. To read a Serial Memory the Chip Select is driven Low, the address is clocked serially into the Serial Input pin then the data is clocked out of the Serial Output pin. Writing to a Serial Memory is similar, but the data is written to the Serial Input pin instead of being read by the Serial Output pin. Serial Memories generally have additional commands and internal buffering to try and reduce the number of Bus Read and Bus Write operations required.

Figure 3. Connection between Microcontrollers and Serial Memories



Serial NAND Memories transfer address and data serially, though they use an 8-bit wide parallel bus. Figure 4 shows an example of a Serial NAND Memory. Address and data are transferred 8-bits at a time from the microcontroller to the Serial NAND Memory and visa-versa. Each operation between the microcontroller and the Serial NAND Memory consists of the command, the address and one or more bytes of data. Serial NAND Memories are not random access, it takes in the order of 10µs to select the first byte in a page to be read, after that subsequent consecutive bytes can be read quickly (in the order of 50ns). Serial NAND Memories are often used in conjunction with simple controllers (e.g. a Flash ATA Controller for PC-cards) rather than connected directly to microprocessors or microcontrollers.

Figure 4. Connection between Microcontrollers and Serial NAND Memories



Connecting Parallel Memories to the System Address Bus

In this section an overview of the connection of a Flash Memory to system address bus will be looked at. STMicroelectronics have a range of application notes dedicated to helping designers connect Flash Memories to specific microprocessors and microcontrollers. Refer to these application notes for more details on how to connect your Flash Memory to a specific microprocessor. If an application note covering the microprocessor you require has not been written yet then the other application notes are worth reading as they contain information that is relevant to all microprocessors.

On Flash Memories with an 8-bit wide data bus the address pins are labelled with the normal conventions, A0, A1... A*n*. A0 selects the byte at an odd or even address. 16-bit wide Flash Memories label the address pins A0, A1... A*n*, but this time A0 selects the *word* at an odd or even address. For a 32-bit wide memory A0 is used to select the *double word* at an odd or even address.

Flash Memories that support two or more bus widths have negative address pin labels. For example a x8/x16 (8-bit or 16-bit width) Flash Memory has address pins A0, A1... A*n* in 16-bit mode and address pins A–1, A0, A1... A*n* in 8-bit mode. The A–1 (“A minus one”) address pin doubles as data bus pin DQ15 in 16-bit mode.

The reason for explaining this in detail is to ensure that the connection to microprocessors is understood. Depending on the configuration chosen, A0 of the Flash Memory will not necessarily connect to A0 of the microprocessor, as the following examples will show. Furthermore, the addresses that the software is required to use when issuing commands to the Flash Memory must be translated from the microprocessor address space to the Flash Memory address space, otherwise the Command Interface will not accept the commands.

Two types of Flash Memory widths will be examined in detail. These are x8 bit Flash Memories and x8/x16 bit Flash Memories. The detail of the other bus widths are similar to these examples. Figure 5 shows the address bus connection of 8-bit Flash Memories to an 8-bit microprocessor and a 16-bit microprocessor.

Figure 5. Connection between 8-bit Flash Memories and microprocessors

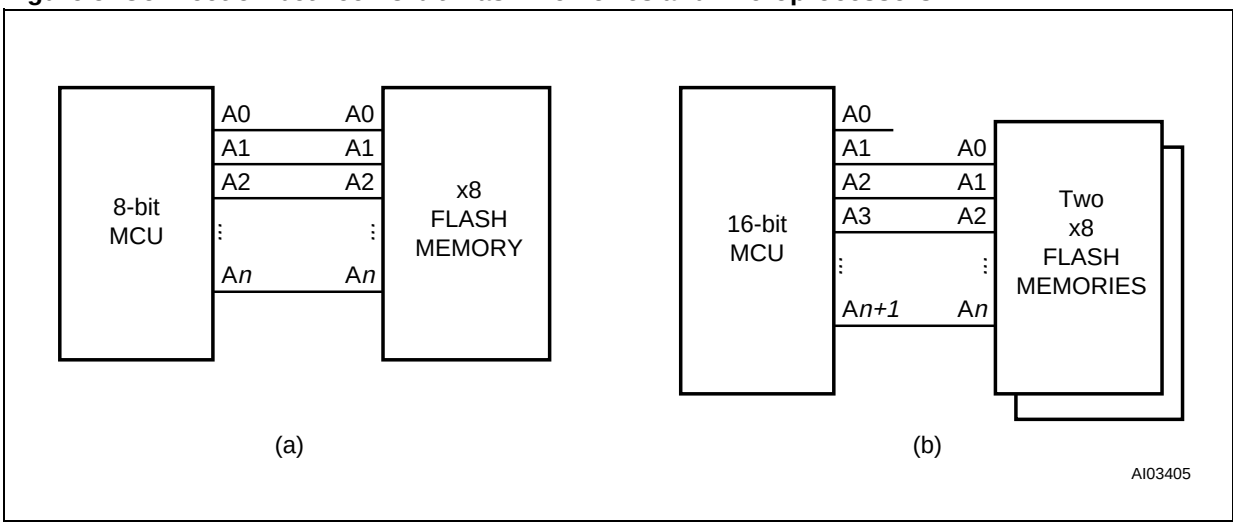


Figure 5(a) shows the connection to the 8-bit microprocessor. Here A0 on the microprocessor connects directly to A0 on the Flash Memory. There is no need to adjust the addresses when issuing commands because the Flash Memory and the Microprocessor share the same address space conventions. In Figure 5(b) there are two 8-bit Flash Memories connected in parallel. One memory holds the Least Significant Byte and the other holds the Most Significant Byte. Now the address bus of the microprocessor is skewed compared to the address bus of the Flash Memory. The software will have to take into account the difference between the Flash Memory’s address space and the microprocessor’s address space when issuing Commands.

Figure 6 shows the address bus connections of x8/x16 bit Flash Memories to an 8-bit and a 16-bit microprocessor.

Figure 6. Connection between x8/x16 bit Flash Memories and microprocessors

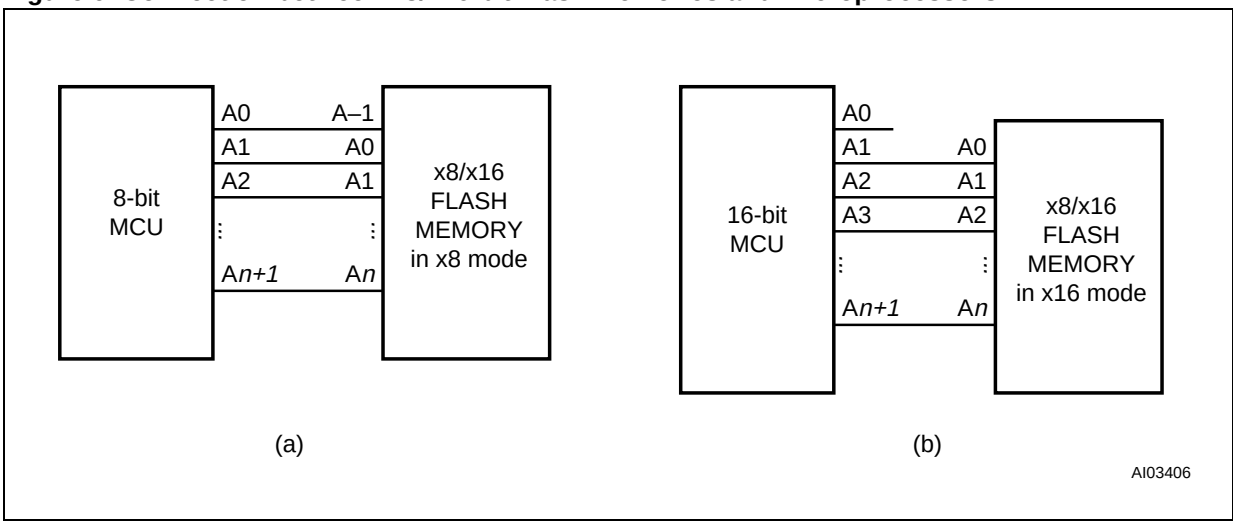


Figure 6(a) shows the connection of an x8/x16 bit Flash Memory in x8 mode to an 8-bit microprocessor. Care must be taken when using the memory in this configuration. The Data Sheet has two tables describing the commands because the address locations for the commands change depending on how the memory is used. This mode is where the address locations for the commands are shifted compared to the 8-bit parts. In Figure 6(b) the Flash Memory is in x16 mode and connected to a 16-bit microprocessor. In this mode the other table in the Data Sheet should be used and address locations for the commands follow the conventions used by 8-bit or 16-bit parts. However, care is still needed in the software because the microprocessor's address pins are skewed compared to the Flash Memory's address pins and A0 of the microprocessor is not used by the Flash Memory.

Second Sourcing

In the group of Parallel Memories made by STMicroelectronics there are two families, the M28 family and the M29 family. The behavior of the memory within the family is almost identical, both between parts and between manufacturers. For example, an M29F400B and an Am29F400B (from AMD) can be used interchangably. One exception to this is the Device Code and Manufacturer Code. Software that relies on the Device Code or Manufacturer Code will need to be changed to use the codes of either part. Moving from an M29F400B to an M29F160B requires little change to the electronics or software; the additional address lines need to be accounted for on the PCB and the software needs to be aware that there is an extra one and a half megabytes of storage space. (Often the software will read the Manufacturer Code and Device Code to recognize the Flash Memory).

Across families it is more difficult to second source parts. The pin connections between M28 family parts and M29 family parts are not identical (though they are as similar as possible). The Command Set accepted by the Command Interface is not the same, software for one family will not work on the other. Whichever family you choose the properties of the memory are very similar: they have similar storage capacities, similar power requirements, similar package sizes, etc.

One major difference between the families is in the erase times. The Block Erase time for each family is similar. However, M29 series Flash Memories can erase multiple blocks in parallel. When several Blocks need erasing an M29 series Flash Memory will win by a long margin. M28 series Flash Memories generally respond faster to Erase Suspend (they have lower Latency), which compensates in some ways for their longer Chip Erase times.

The Common Flash Interface (or CFI) was intended to avoid compatibility problems. The interface defines the architecture of the Blocks in the Flash Memory, the timing parameters, Command Set and other features that software needs to know. Be aware, however, that having a Common Flash Interface Flash Mem-

ory and Common Flash Interface software does not mean that they will communicate, it just means they will be able to tell you that they cannot communicate. Some Common Flash Interface software does support both families and this can be used to make the parts truly second source across families.

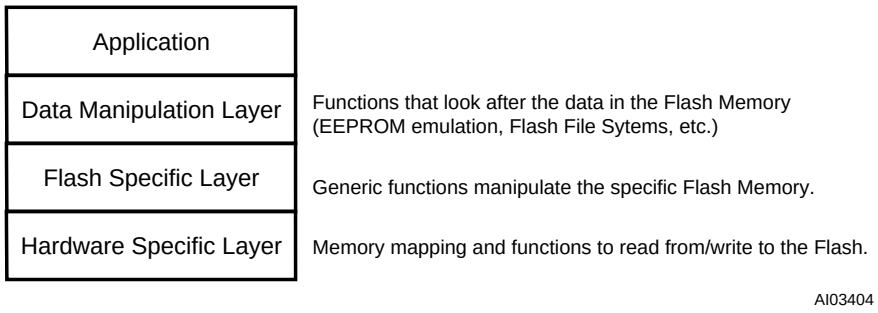
SOFTWARE ISSUES

STMicroelectronics provides software drivers for nearly all of its Flash Memories. If the driver you require if not available then contact STMicroelectronics, the driver for the part you require may be in development and become available shortly.

Some of the details that software engineers should be aware of have been described in the Hardware Issues section. These include the shifting of the microprocessor address bits compared to the Flash Memory address bits. The address locations of the commands are affected by the way that the Flash Memory's address bus is connected to the microprocessor's address bus.

In all but the simplest of software models designers use layers to hide the detail of different parts of a design from other parts. Software written for Flash Memories is no exception to this. Different designs use different numbers of layers to abstract the manipulation of the data. In this application note four layers will be looked at: Figure 7 shows the layers that will be described.

Figure 7. Layers of Software Abstraction used in conjunction with Flash Memories



In the Hardware Specific Layer we find functions that are used to communicate with the Flash Memory over the microprocessor bus. In general the microprocessor bus only has two operations, Bus Read and Bus Write. In the Flash Memory drivers provided by STMicroelectronics the user has to write two functions, **`FlashRead()`** and **`FlashWrite()`** to allow the functions in the next layer up to talk to the Flash Memory in their system.

The Flash Specific Layer translates from standard functions into operations the Flash Memory will execute. The standard functions that a Flash Memory provides are Auto Select, Program, Block Erase, Chip Erase, etc. It does not matter which Flash Memory you have, you will be able to execute these operations. In the Flash Memory drivers provided by STMicroelectronics these standard functions are provided by the C functions **`FlashAutoSelect()`**, **`FlashProgram()`**, **`FlashBlockErase()`** and **`FlashChipErase()`**. If you are using the drivers then you do not need to know how to program the Flash Memory, you only need to call **`FlashProgram()`** and this layer will take care of the rest.

In the Data Manipulation Layer users are able to update data stored in the Flash Memory without worrying about how this is done. The user will want to issue commands like "update picture 1 with these changes" or "modify my address book database to include a new address". Because Flash Memories are erased in Blocks and the picture and the database may be stored in the same Block, care needs to be taken when updating one of them. The Block cannot be erased to update the picture without also saving the database. The Data Manipulation Layer takes care to ensure that the data is kept correctly and securely all the time.

The usual technique is to divide each Block into smaller partition and keep track of which partitions (and what order of partitions) store the picture, the database, and any other data items. There are several commercially available software programs to perform the Data Manipulation Layer.

The Data Manipulation Layer is the subject of Application Note AN1188. This application note deals with several techniques of data storage, including EEPROM emulation and linked lists. (EEPROM emulation is useful for people who are upgrading from using an EPROM and an EEPROM to a single Flash Memory).

Flash Memories are now fast enough to enable microprocessors to run their programs directly from the Flash Memory, rather than copying the code to another memory type, such as DRAM. Running your software directly from the Flash Memory has some drawbacks that need to be explained.

When you take a Flash Memory out of Read Mode it no longer responds with the data stored in the memory array. If you are running your software from the Flash Memory and you issue a program or erase command then your software is likely to die horribly. The next instruction read from the Flash Memory will read the Status Register, not the op-code for the microprocessor.

There are two solutions to this problem. Some Flash Memories are available with two Banks (known as Dual Bank Flash Memories). Program and erase operations on one Bank do not affect the other Bank, so you can run your software from one Bank while programming the other. This is the easiest solution to use, but Dual Bank Flash Memories cost more than those with only one Bank. Internally Dual Bank Flash Memories have 30% to 50% more circuitry, mainly because they require two read paths from the memory array (they are nearly two Flash Memories on one silicon die).

The alternative solution is to copy the software required to program or erase to another memory and run it from there. Not all of the software needs to be copied, only the part that executes the program or erase command. The disadvantage of this method though is that, if an interrupt occurs and your interrupt service routine is in the Flash Memory, you will not be able to access it until the Flash Memory completes its operation. In real time systems that cannot tolerate the microprocessor being busy for one or more seconds this is clearly a problem.

The Erase Suspend command is useful for temporarily recovering your system during Block Erase operations. If an interrupt occurs while the Flash Memory is erasing you can issue an Erase Suspend instruction. The Flash Memory will take about 10µs to pause the erase operation and return to Read Mode. Flash Memories with Low Latency respond much faster, within about 1µs. Once the Flash Memory has returned to Read Mode the interrupt can be run. Be sure to check that the Flash Memory has entered Read Mode by reading the Status Register. Once the interrupt has run the Erase Resume command will restart the erase operation where it left off.

CONCLUSION

Many of the issues to be considered when choosing a Flash Memory have been addressed in this Application Note. Flash Memories a are rugged, compact, easy to use data storage solution for electronic products. Suitable for low power applications they are finding their way into all sorts of modern consumer products.

The types of data stored in the Flash Memory should be mapped at the outset of the design process. Boot or Uniform Architecture Flash Memories can easily be chosen from considering the data types to be stored. The size, voltage, speed, package, etc. can be chosen to suit the application. Other quantities such as power consumption are fixed. Software drivers to command the Flash Memory are available from ST-Microelectronics, software to manipulate the data is available from a variety of suppliers.

STMicroelectronics provides a range of different Flash Memories to suit every application. Take the plunge today and choose a Flash Memory in your next application!

GLOSSARY OF TERMS USED IN FLASH MEMORIES

The terms in this glossary are intended to help newcomers to Flash Memories understand the jargon associated with Flash Memories. Not all of the terms in this glossary will apply to all Flash Memories.

3 Hex Codes. The Command Interface of some Flash Memories will not accept commands directly, instead it is necessary to issue some Coded Cycles to unlock the Command Interface before issuing the command. The purpose of the Coded Cycles is to protect the contents of the Flash Memory from power on glitches, power off glitches and faulty software. Each Coded Cycle requires the correct data on the data bus and the correct address on part of the address bus. The upper bits of the address bus are ignored during the Coded Cycles. Recently the number of address bits that needed to be correct (i.e. were not ignored) changed from A0-A14 to A0-A10. A Flash Memory that only requires address bits A0-A10 to be correct for the Coded Cycles is said to need “3 Hex Codes”.

Block. *Also known as “Sector”.* A Block is a group of Flash Memory cells that have to be erased together. Usually Blocks are 64 Kbytes in size, but other sizes exist (Boot Blocks, Parameter Blocks). Once a byte (or word) has been programmed it is necessary to erase the entire Block containing the byte before the byte can be programmed again. All of the bytes in the Block will be erased at the same time, so it is important to save their data values before erasing them.

Block Erase Time. *Also known as “Sector Erase Time”.* This is the time it takes to erase a Block. On some memories several Blocks can be erased at the same time. It is faster to erase two Blocks at the same time than to erase one then the other. However, erasing two Blocks still takes longer than erasing one Block (mainly due to the time it takes the Program/Erase Controller to program all of the cells to ‘0’ before erasing).

Block Protection. *Also known as “Sector Protection”.* Blocks can be protected against accidental program or erase operations. If a Block is protected then it will not be possible to change the data in the Block. Details on how to protect and unprotect Blocks are given in Application Note 1122.

Block Protection Status. *Also known as “Sector Protection Status”.* The Auto Select mode of a Flash Memory can be used to determine if a Block is protected or unprotected. Protected Blocks cannot have their data contents changed. In order to change the contents of a protected Block it is necessary to unprotect the Block, either temporarily using the Reset/Block Temporary Unprotect pin or permanently in a Flash Programmer.

Boot Block. *Also known as “Boot Sector”.* A Boot Block is a Block of memory that is usually smaller than the normal Block size. The Boot Block(s) will always be situated at the top or bottom of the memory (Top Boot Block or Bottom Boot Block). The purpose of the Boot Block is to store software code that the microprocessor boots from. Generally the Boot Code does not change very often, whereas the Application Code may change regularly. By placing the Boot Code in one or more Boot Blocks it can be protected against accidental program or erase operations that could corrupt it. Often Boot Code will include some mechanism to replace the Application Code so that, if the Application Code becomes corrupt, it will be possible to recover the system without removing the Flash Memory and reprogramming it in a Flash Memory Programmer.

Boot Sector. See Boot Block.

Bus Read Operation. The Flash Memory Data Sheets use the term Bus Read Operation and Bus Write Operation to avoid mis-understandings that can arise with the term “read” or “write”.

Bus Write Operation. The Flash Memory Data Sheets use the term Bus Read operation and Bus Write operation to avoid mis-understandings that can arise with the term “read” or “write”. The term “write” can be particularly mis-leading because the phrase “writing to the memory” could be interpreted as “writing to the Command Interface” or “programming the memory”.

$\overline{\text{BYTE}}$. See Byte/Word Organization Select pin.

Byte/Word Organization Select pin. *Also known as "BYTE".* When the Byte/Word Organization Select pin is High, V_{IH}, the Flash Memory has a word-wide (x16) data bus. When it is Low, V_{IL}, the Flash Memory has a byte-wide (x8) data bus. The pin DQ15A–1 changes operation from an address pin (A–1) and a data pin (DQ15) when Byte/Word Organization Select changes.

CFI. Abbreviation for Common Flash Interface.

Coded Cycles. To increase the security of the data stored in Flash Memories the Command Interface will only accept commands that are preceded by the correct Coded Cycles. The Coded Cycles prevent the Flash Memory from losing data due to faulty software, power on and power off glitches.

Command Interface. The Command Interface is the internal state machine in the Flash Memory that interprets the Bus Write operations. Bus Write operations do not directly affect the data stored in the Flash Memory; instead, program and erase commands are required to change the memory's data. The Command Interface will verify that the commands are valid and change the mode of the Flash Memory accordingly. Program and erase commands are executed by the Program/Erase Controller.

Common Flash Interface (CFI). The Common Flash Interface is a standard defined to try and improve the second-source nature of Flash Memories. The Common Flash Interface is stored in an additional part of the memory that can be accessed by issuing the CFI Query Command. Information such as the memory size, Block sizes and addresses, program and erase timings, Command Set, etc. are stored in the Common Flash Interface memory space. The Common Flash Interface is useful for PCMCIA cards since it allows the software on the host to identify the configuration of the memory in the PCMCIA card. In general however, the Common Flash Interface does not provide many of the benefits that users expect. Having a Common Flash Interface compatible Flash Memory and Common Flash Interface compatible software does not guarantee that the system will work.

Command Set. Not all Flash Memories respond to the same commands, there are two different Command Sets. STMicroelectronics manufactures two different types of Flash Memories, M28 series and M29 series, one series for each Command Set. The basic functionality of each Flash Memory series is the same, they have fast random access times, they are reliable, they can be used for both code and data storage. The differences are in the Command Set and the pin configurations. If you are looking for a second-source to your STMicroelectronics Flash memory then Intel provide Flash Memories compatible with the M28 series and AMD or Fujitsu are compatible with the M29 series.

Device Code. See Electronic Signature.

Dual Bank. *Also known as "Dual Plane".* Dual Bank Flash Memories have their Blocks divided between two Banks. When program or erase operations are taking place in one Bank, read operations addressed to the other Bank read from the memory array; read operations addressed to the Bank where the program or erase operation is taking place will read the Status Register. The advantage of a Dual Bank Flash Memory is that the Application Code can be run directly from the Flash Memory while program or erase operations are taking place. Flash Memories that do not have two or more Banks require the Application code (or at least part of it) to be copied to another memory (such as DRAM) during program or erase operations.

DQ15A–1. *(Read DQ15A minus 1).* The DQ15A–1 pin is a special pin on Flash Memories that can be configured for byte or word data bus sizes. When the Flash Memory is accessed in byte mode (the Byte/Word Organization Select pin, $\overline{BYTE}$, is Low, V_{IL}) DQ15A–1 is an input address pin; if V_{IL} is applied to DQ15A–1 during a read operation then the byte output is the value that would be output on DQ0 to DQ7 in word mode; if V_{IH} is applied to DQ15A–1 during a read operation then the byte output is the value that would be output on DQ8 to DQ15 in word mode. When the Flash Memory is accessed in word mode (the Byte/Word Organization Select pin, $\overline{BYTE}$, is High, V_{IH}) DQ15A–1 is a data pin.

Dual Plane. See Dual Bank.

Electronic Signature. The Electronic Signature of a Flash Memory consists of the Manufacturer Code and Device Code. The Manufacturer Code for STMicroelectronics is 20h; each type of Flash Memory made by STMicroelectronics has a different Device Code, for example the M29F040B has a Device Code of E2h. The Electronic Signature can be read through special bus operations, by using the Auto Select Command or by using the Read Electronic Signature Command.

Embedded Algorithm. The Embedded Algorithms are executed by the Program/Erase Controller. The algorithms determine the sequence of pulses to apply to each cell or Block, ensuring that the Flash Memory is programmed or erased correctly.

Erase Suspend. Because it takes a long time (in the order of seconds) to erase it is often useful to be able to suspend the Block Erase operation and read data from another Block. The Erase Suspend command can be executed at any time during an Block Erase operation. Data from Blocks that are not being erased can be read and, on some Flash Memories, data can be programmed too. The erase operation can be resumed using the Erase Resume Command.

Hardware Reset. A Hardware Reset is executed when the power is applied to the Flash Memory or when the Reset/Block Temporary Unprotect pin is Low. A Hardware Reset leaves the Flash Memory in Read Mode where the data in the Flash Memory can be read like a ROM or EPROM. Hardware Reset operations during program or erase operations should be avoided since they can leave the contents of the affected memory cells in an indeterminate state. A Hardware Reset will always stop any operations on the Flash Memory and return to Read Mode; by contrast Software Resets (through the Read/Reset command) will not abort all operations.

Identification Voltage. *Also known as "V_{ID}".* The Identification Voltage is the voltage that should be applied to some of the pins in the Flash Memory (usually A9) to read the Electronic Signature (Device Code or Manufacturer Code). The Data Sheets describe reading these codes in the Special Bus Operations section. The Identification Voltage is used by Flash Programmers to find out what type of Flash Memory is fitted automatically, the user does not need to select the device in Flash Programmers that use this feature. The Identification Voltage is also used for other purposes; raising the Reset/Block Temporary Unprotect pin to the Identification Voltage temporarily unprotects all of the Protected Blocks so their content can be changed; the Identification Voltage is also required to protect a Block or unprotect all Blocks.

Main Block. *Also known as "Main Sector".* There are usually more Main Blocks in a Flash Memory than any other type of Block. They form the bulk of the memory size. Usually Main Blocks are 64 Kbytes (32 Kwords) in size. Main Blocks are typically used for code storage, data storage, flash file systems, etc.

Main Sector. See Main Block.

Manufacturer Code. See Electronic Signature.

Parallel Memory. The term Parallel Memory is used to describe the access mechanism for getting data in and out of the memory. A Parallel Memory has fully parallel address and data buses (or a fully parallel multiplexed address/data bus). Several control lines (Chip Enable, Output Enable and Write Enable) indicate the type of access (Bus Read or Bus Write). This is the method used to map the memory to an address in the microprocessor address space. Parallel Memories have fast random access to the data in the memory. Other access mechanisms include Serial Memories (with I^2C or SPI interfaces) and Serial NAND Memories.

Parameter Block. *Also known as "Parameter Sector".* A Parameter Block is a Block that is smaller than the normal Block size for the Flash Memory. Parameter Blocks are useful for storing user parameters in the Flash memory. Their small size is designed to be large enough to hold all the user parameters, but, because they are smaller than the Main Blocks they erase faster, giving a better response time to user requests. Usually Parameter Blocks appear in pairs. When the user changes one or more values it is a simple procedure to copy the data from one Parameter Block to the other, changing the data as it is copied. The old Parameter Block can then be erased. A flag can be used to mark the most recent Parameter Block.

Parameter Sector. Also see Parameter Block.

P/E.C. Abbreviation for Program/Erase Controller.

Program/Erase Controller. The Program/Erase Controller is responsible for programming the contents of a Flash Memory cell and erasing Blocks. To program a cell correctly the Program/Erase Controller has to pulse each cell until the state of the cell is within the correct margins. Too many pulses damage the cell, reducing the number of Program/Erase Cycles the Block can survive. Erase operations have to program all of the cells so they are in the same state (their contents will be '0') before erasing all of the cells in one go. If the cells are not all programmed to the same state then damage can occur during the erase operation. The number of Program/Erase Cycles that a Flash Memory can tolerate depends, in part, on the Program/Erase Controller executing its pulses correctly.

Program Time. This is the time it takes to program a byte or a word. Typically it takes between 8µs and 10µs to program. On STMicroelectronics memories programming a byte takes the same time as programming a word; it is worthwhile programming by word if you have a lot of addresses to program.

Reset/Block Temporary Unprotect pin. *Also known as "$\overline{RP}$".* The Reset/Block Temporary Unprotect pin can be used to cause a Hardware Reset to the Flash Memory or Temporarily Unprotect all of the Protected Blocks. Not all Flash Memories have this pin. The pin should be Low, V_{IL}, when the microprocessor is reset. To cause a Hardware Reset the pin should always be held Low for at least t_{PLPX}. During normal operation the pin should be High, V_{IH}. To Temporarily Unprotect all of the Protected Blocks (thereby allowing changes to their content to be made) the pin should be raised to the Identification Voltage, V_{ID}.

$\overline{RP}$. See Reset/Block Temporary Unprotect Pin.

Sector. See Block.

Sector Erase Time. See Block Erase Time.

Sector Protection. See Block Protection.

Sector Protection Status. See Block Protection Status.

Serial Memory. The term Serial Memory is used to describe the access mechanism for getting data in and out of the memory. Serial Memories use a serial bus, such as SPI, to communicate with the memory and transfer the data to and from the memory. Other access mechanisms include Parallel Memories and Serial NAND Memories.

Serial NAND Memory. The term Serial NAND Memory is used to describe the access mechanism for getting data in and out of the memory. The term NAND is the technology used in the Flash Memory cell that stores the data. NAND memories are read, programmed and erased one page at a time. It takes in the order of 10µs to select the page in the memory, after that the data can be read with an access time in the order of 50ns. The bus is usually 8 bits wide with some extra control signals. The address and data are written and read on the same 8 bits. Other access mechanisms include Parallel Memories and Serial Memories.

Small Main Block. *Also known as "Small Main Sector".* The Small Main Block(s) in a Flash Memory are a result of the Top or Bottom Block being divided into Boot Blocks and Parameter Blocks. The Small Main Block usually occupies the remaining space. If the memory is being used for code storage then the Application Code can start in the Small Main Block and continue into the next Main Block(s).

Small Main Sector. See Small Main Block.

Software Reset. A Software Reset is executed by issuing the Read/Reset command. After a Software Reset the Flash Memory will return to Read Mode, unless a program or erase operation is taking place. Software Resets during program and erase operations should be avoided since they can leave the Flash Memory cells in an indeterminate state. Different Flash Memories behave differently when a Software Reset is issued during program or erase operations, some ignore the command, others will abort the operation. Refer to the Data Sheet of each Flash Memory for information on how it behaves. Hardware Resets, by contrast, will always abort any program or erase operation.

Status Register. The Status Register reports the progress of a program or erase operation. The Status Register is used to identify if a program or erase operation is still taking place, or if it has completed. Error bits in the Status Register are used to identify if an error has occurred during program or erase operations. M28 series Flash Memories and M29 series Flash Memories have different Status Register bit definitions. See the Data Sheets for information on the definitions of the Status Register Bits.

Temporary Block Unprotect. *Also known as "Temporary Sector Unprotect".* Some memories have a Reset/Block Temporary Unprotect pin ($\overline{RP}$) that, when raised to the Identification Voltage, V_{ID}, allows all protected blocks to be programmed or erased. When the pin returns to normal CMOS voltage levels the protected blocks can no longer be changed.

Temporary Sector Unprotect. See Temporary Block Unprotect.

Three Hex Codes. See 3 Hex Codes at the start of the glossary.

Uniform Block. *Also known as "Uniform Sector".* A Flash Memory will be described as a Uniform Block Flash Memory if all of the Blocks in it are the same size. For example, the M29F040B is a Uniform Block Flash Memory because all of the Blocks are 64 Kbytes in size.

Uniform Sector. Also see Uniform Block.

Unlock Bypass Command. The Unlock Bypass Command is used to reduce the number of Bus Write operations that are required to program a Flash Memory. Some Flash Programmers have long access times (due to their flexibility, they need to set each pin individually). When programming a Flash Memory considerable time can be spent writing the Coded Cycles before each Program Command. To avoid needing to do this and speed up the programming of the Flash Memory the Unlock Bypass command can be issued, reducing the number of Bus Write operations per Program command from 4 to 2. Speed increases of 30% can be obtained using this technique. For embedded systems with fast access times there is little benefit in using the Unlock Bypass Command.

V_{IH}. V_{IH} is the definition of a High voltage or logic level '1'. The Data Sheet defines its range of values in the DC Characteristics.

V_{IL}. V_{IL} is the definition of a Low voltage or logic level '0'. The Data Sheet defines its range of values in the DC Characteristics.

If you have any questions or suggestion concerning the matters raised in this document please send them to the following electronic mail address:

ask.memory@st.com (for general enquiries)

Please remember to include your name, company, location, telephone number and fax number.