Migrating between Boot and Uniform Sectored Flash Devices

Application Note

July 2003

Publication Number **22374** Revision **A** Amendment **0** Issue Date **May 12, 1999**

Migrating Between Boot and Uniform Sectored Flash Devices

Application Note

The purpose of this bulletin is to describe the software algorithm changes necessary to convert from a boot sectored flash device to a uniform sectored flash device or vise versa. Changes are required only for applications that require sector erase capabilities in system. If the flash device is read-only within a system, programmed in system but not erased, or is erased with chip erase only, there is no functional difference between boot and uniform sectored devices and therefore no need to make any software changes.

The Origins of Sector Erase Flash Memories

The first Flash devices manufactured were bulk erase Flash devices. Bulk erase Flash devices could program (change a "1" to a "0") information into the device a byte at a time, but required the entire Flash device be erased (change "0's" to "1's") every time a byte of memory needed to be erased. This wasn't the most efficient way of changing data because you had to erase and reprogram the whole flash device for any erase no matter how small the change. To solve this problem, flash with erasable sectors was developed. Now, instead of erasing the entire flash device at one time, you could erase all memory within an erase sector independent of other sectors. This was a 16 Kbyte block of memory to begin with, so the 1 Mb device (where erase sectors were first implemented) had eight 16 Kbyte erase sectors which could be erased independently of each other. Since this device was introduced, erase sectors have further changed into different organizations to better suit the needs of today's applications. Flash devices now are organized into two different erase architectures, boot and uniform, and erase sectors also have the ability to be individually or group protected against accidental programming or erase in system.

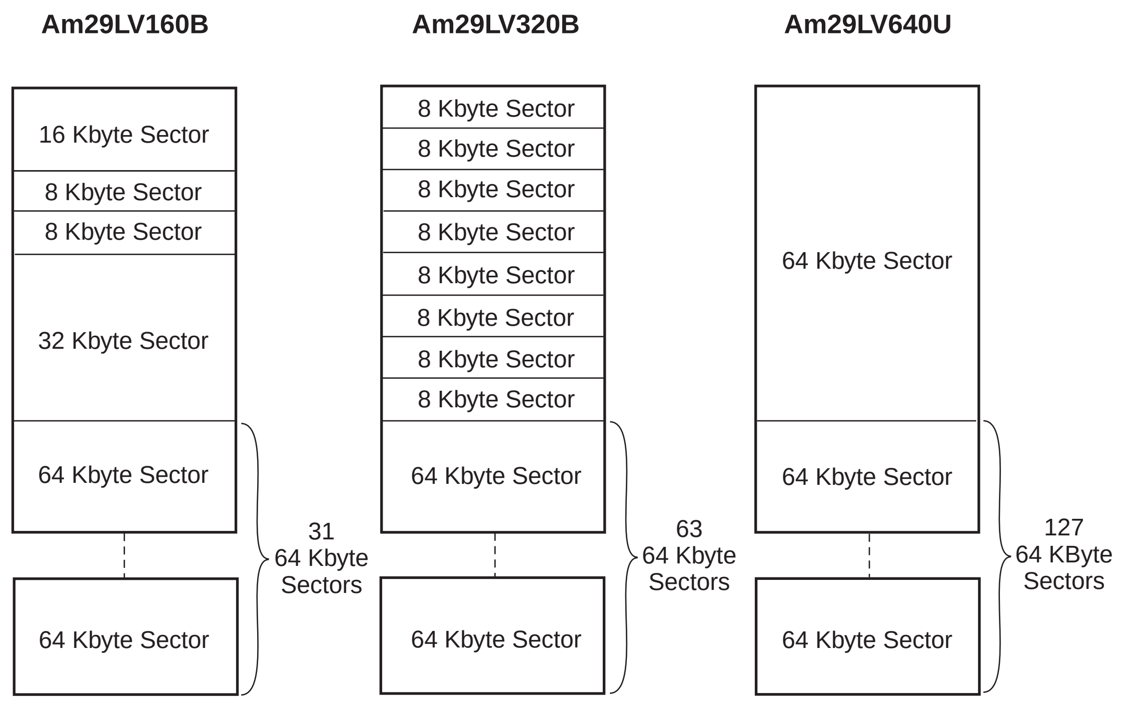
The Difference Between Boot and Uniform Sectored Flash Architectures

The difference between a boot sectored device and a uniform sectored device is the make up of the first or last 64 Kbytes of memory. For a uniform device, the 64 Kbytes are a single erase block, which is similar to the rest of the erase sectors within the device. Uniform sectored devices are ideal for building large memory arrays since the erase blocks are all the same size and therefore easily managed. For boot devices, the first or last 64 Kbytes is subdivided into several erase blocks. Typically these smaller erase blocks are in one of two configurations: 16 Kbyte, 8K byte, 8K byte, 32 Kbyte or eight 8 Kbyte sectors.

These smaller erase sectors are generally used to store boot code or parametric data. The reason for this is that boot code and parametric data usually does not require a whole 64 Kbyte sector, but still need to be placed in a separate sector so it can be properly protected or changed without affecting other portions of code. Smaller boot sectors permit this without having to waste an entire 64 Kbyte sector. Boot sectored devices also have the option of placing the smaller boot sectors within the first 64 Kbytes of the address range (called bottom boot) or last 64 Kbytes of the address range (called top boot). The reason for this option is due to the way microcontrollers access the initial boot code required to power on. Two conventions for microcontrollers exist: top-boot convention, where the first address fetch on re-initialization is performed at the top of memory space, and bottom-boot, where the first fetched address is at the bottom of memory space. Microcontrollers based of IBM, RISC, or Motorola processors boot from bottom of memory space, and those from Intel and DEC boot from the top of memory space.

The following are examples of AMD's low voltage 16 Mbit and 32 Mbit boot sectored devices and low voltage 64 Mbit uniform sectored device.

Publication# **22374** Rev: **A** Amendment/**0**
Issue Date: **May 12, 1999**



Software Changes When Migrating From Boot To Uniform

Case 1

The system does not require modifications of any of the boot sectors independent of one another, i.e. all boot sectors can be erased together.

Solution

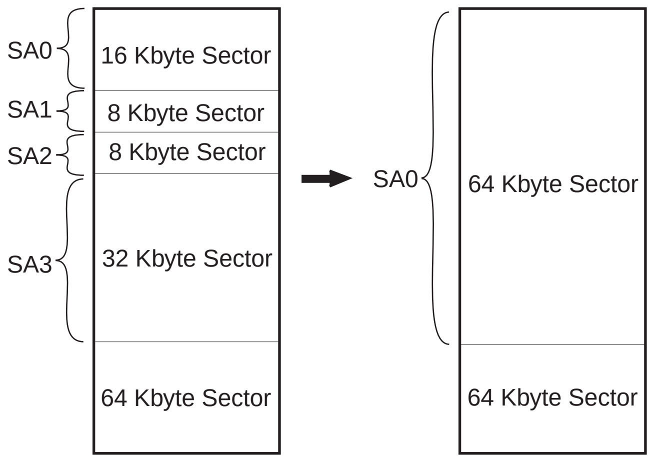
A small software change may be required. Instead of asking the device to erase all of the smaller boot sectors, you can just ask it to erase the single larger 64K byte block. The command sequence to erase all of the boot sectors at once for the Am29LV160BB should look something like this:

Address	555	2AA	555	555	2AA	SA0	SA1	SA2	SA3
Data	AA	55	80	AA	55	30	30	30	30

If the boot command sequence is used on a uniform device the net effect will be the same, the first 64 Kbytes of the flash device, which SA0, SA1, SA2, & SA3 now point to, will be erased once. The uniform flash device will ignore the additional sector erase commands since the commands essentially tell the flash device to erase the first uniform sector four times. No code change is necessary, however the code will not be as clean as if the code had specifically been written for a uniform sectored device. In order to clean up the code, just remove the last three sector erase pulses. The modified sector erase command should look like the following sequence:

Address	555	2AA	555	555	2AA	SA0
Data	AA	55	80	AA	55	30

where SA0 is now any address within the first 64 Kbyte uniform sector.



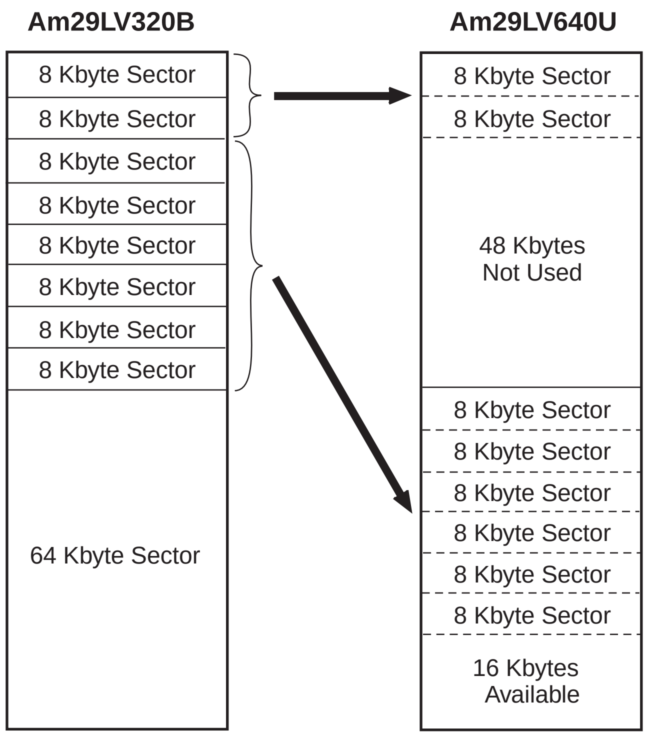
Old Boot Sectored New Uniform Sectored

Case 2

The system requires an erase to one or more of the boot blocks, but data in the other boot blocks need to be preserved.

Solution

You will need to change some address pointers for the smaller boot blocks. To determine if the code currently in the boot sectors needs to be placed in its own individual 64 Kbyte sector, the question you need to ask is: Does the code need to be modified by itself or can it be updated at the same time as other parts of the code? If it can be modified with other portions of the code, then you can group portions of the code together to potentially occupy all or part of a single 64 Kbyte sector. If it absolutely needs to be changed by itself then you must place it in its own 64 Kbyte sector. For each additional sector required you need to change the address pointers. Let's look at an example:



In this case, let's assume that you are migrating from a 32 Mb boot block device to a 64 Mb uniform device. The first two 8K bytes sectors are the boot code, which is never changed, and the other 8 Kbyte boot sectors are used to store the operating code, which may need to be upgraded later. The boot code must be loaded into it's own 64 Kbyte sector and the addressing does not change. The rest of the boot sectors need to be placed in a one of the new 64 Kbyte sectors available in the larger memory device and the address pointers updated. Also please note that when migrating from boot to uniform it may be necessary to leave a small amount of memory space unused (48 Kbyte in the last figure) in order to retain the flexibility of the older boot sectored device.

Software Changes When Migrating From Uniform To Boot

Case 1

The system does not need to take advantage of the smaller boot sectors.

Solution

A small software change is needed to tell flash device to erase all of the smaller sectors at once. This can be done by adding the sector addresses to the end of the normal sector erase command. Looking at an example going from uniform to bottom boot, the original sector erase command for a uniform sectored device would look like this:

Address	555	2AA	555	555	2AA	SA0
Data	AA	55	80	AA	55	30

Where SA0 is any address within the first 64 Kbyte sector. In order to convert to a bottom boot device you need to modify the sector erase command to look like this:

Address	555	2AA	555	555	2AA	SA0	SA1	SA2	SA3
Data	AA	55	80	AA	55	30	30	30	30

Now SA0 points to any address within the first 16 Kbytes, SA1 points to an address within the next 8 Kbytes, SA2 points to the next 8 Kbytes after that, and finally SA3 points to the last 32 Kbytes. Together the sector erases cover the first 64 Kbytes of the device. So this command will erase all of the small boot sectors at once, effectively making it one large uniform sector. All other sector erase commands will remain the same since the devices are identical past the first 64 Kbytes.

Case 2

The system will be taking advantage of the flexibility offered by the smaller boot sectors by erasing them separately of one another.

Solution

Software changes will be necessary, but only to control erasing of the smaller sectors. Erasing of other sectors, programming, and reading the flash device remain the same. For each sector to be erased separately you need to include the following command sequence to specify it independent of the other sectors as well as identifying when the command will be called in the code.

Address	555	2AA	555	555	2AA	SA
Data	AA	55	80	AA	55	30

So with the older uniform sector you need only one erase command for the first or last 64 Kbyte sector. Now with a boot device you can have up to four different erase commands for a 16 Kbyte, 8 Kbyte, 8 Kbyte, and 32 Kbyte configuration or up to eight different sector erase commands for an eight 8 Kbyte boot configuration.

Please note that if a boot sectored device is accidentally used with code designed for a uniform device there could be a potential problem. Some data that was thought to be erased, will still be present (since a sector erase issued to sector 0 will erase 64 Kbyte in the Am29LV017B will only erase a 16 Kbyte sector on the Am29LV320DB). Any attempt to program data from a “0” to a “1” in a sector which has not been erased will result in the flash device going to an error state.