

# Understanding Page Mode Flash Memory Devices

Application Note



July 2003

The following document refers to Spansion memory products that are now offered by both Advanced Micro Devices and Fujitsu. Although the document is marked with the name of the company that originally developed the specification, these products will be offered to customers of both AMD and Fujitsu.

## **Continuity of Specifications**

There is no change to this document as a result of offering the device as a Spansion product. Any changes that have been made are the result of normal documentation improvements and are noted in the document revision summary, where supported. Future routine revisions will occur when appropriate, and changes will be noted in a revision summary.

## **Continuity of Ordering Part Numbers**

AMD and Fujitsu continue to support existing part numbers beginning with "Am" and "MBM". To order these products, please use only the Ordering Part Numbers listed in this document.

## **For More Information**

Please contact your local AMD or Fujitsu sales office for additional information about Spansion memory solutions.

Publication Number **23711** Revision **A** Amendment **0** Issue Date **May 25, 2000**





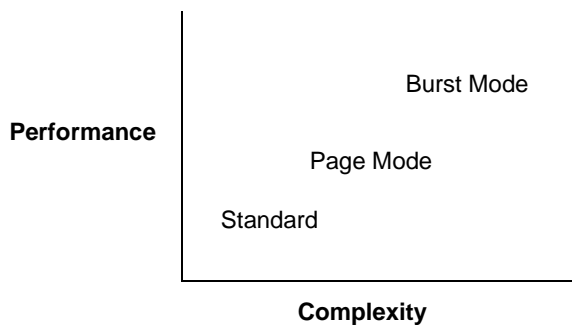
# Understanding Page Mode Flash Memory Devices

## Application Note

Current AMD flash memory products operate with random access times ranging anywhere from 45 ns to 150 ns. Though most applications use an AMD flash device with a random access time of about 70 ns to 90 ns. The faster access times of 45 ns to 55 ns are only available today in lower density devices from 1 to 4 Mbits. However, many new applications need high-speed access, high density, and are migrating to lower voltages to save power. But, higher density and lower voltage tend to reduce performance in a standard random access memory architecture. Therefore, AMD is using different architectural approaches to increasing performance in several new Flash memories. One approach is called *Page Mode*. To help describe how a Page Mode device works, the Am29PL160C will be used as an example.

### DEVICE MIGRATION

Just as the flash memory overall performance increases, the internal architecture also becomes more complex. By comparing the three current AMD flash architectures, Standard, Page Mode, and Burst Mode, you can see how the complexity increases in order to achieve faster access times. The following graph shows a relative performance vs. complexity relationship.



A Page Mode device is pin compatible with a Standard flash device. This ensures an easy migration for a better performance part, with little added complexity to the

overall system. If more performance is still desired, Burst Mode devices can offer that performance. But, the system complexity becomes much greater.

### WHAT IS A “PAGE?”

A page is a small group of memory words that are accessed, internal to the memory, in parallel rather than one at a time. The time to reach the first word in the group is called the initial access time and is analogous to standard architecture Flash memory access times. However, since all the words in the group are stored in an internal buffer following the initial access time, other words in the group can be delivered with a much reduced access time.

All pages in a memory device are the same size, but the size of the pages varies depending on the device. The page size of the Am29PL160C device is 8 words, or 16 bytes, with the appropriate Page being selected by the higher address bits A3–A19. The LSB bits A0–A2 (in word mode) and A–1 to A2 (in byte mode) select the specific word/byte within a page. The pages are therefore always aligned on an 8-word address boundary. Table 1 shows the words for a given page, and Table 2 shows the bytes for a given page.

Table 1. Word Mode

Word	A2	A1	A0
Word 0	0	0	0
Word 1	0	0	1
Word 2	0	1	0
Word 3	0	1	1
Word 4	1	0	0
Word 5	1	0	1
Word 6	1	1	0
Word 7	1	1	1

**Table 2. Byte Mode**

Byte	A2	A1	A0	A-1
Byte 0	0	0	0	0
Byte 1	0	0	0	1
Byte 2	0	0	1	0
Byte 3	0	0	1	1
Byte 4	0	1	0	0
Byte 5	0	1	0	1
Byte 6	0	1	1	0
Byte 7	0	1	1	1
Byte 8	1	0	0	0
Byte 9	1	0	0	1
Byte 10	1	0	1	0
Byte 11	1	0	1	1
Byte 12	1	1	0	0
Byte 13	1	1	0	1
Byte 14	1	1	1	0
Byte 15	1	1	1	1

must go to  $V_{IL}$ , and a valid address must be placed on A19:A0. This first read has an access time  $t_{CE}$  or  $t_{ACC}$  which is typical of a standard flash device. However, a subsequent Page read access to a location anywhere within the same page is much faster. This access time is denoted as  $t_{PACC}$ . Fast Page mode accesses are obtained by keeping A3–A19 constant and changing A0–A2 to select the specific word, or changing A–1 to A2 to select the specific byte, within that page.

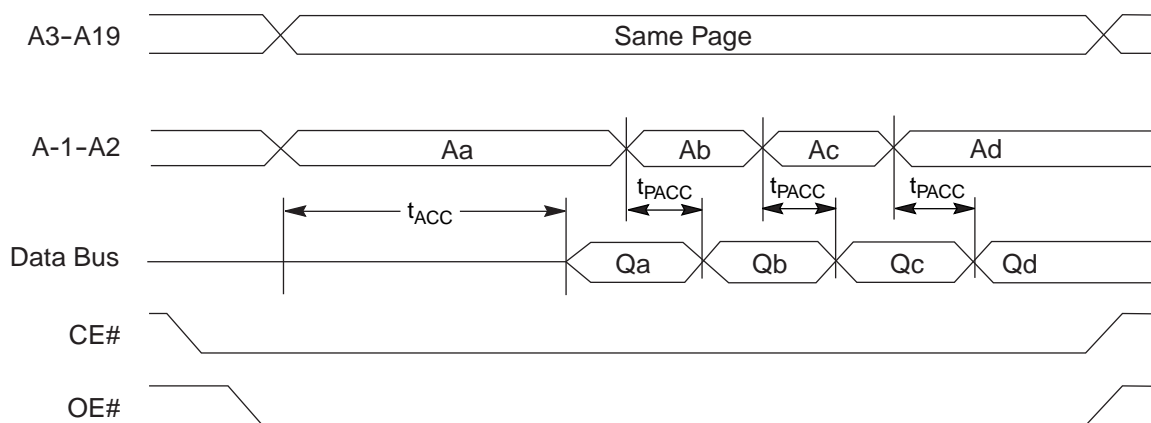
### WHAT ARE THE BENEFITS OF OPERATING IN PAGE MODE?

The major benefit of operating in Page Mode is the speed. Some other benefits are potential reductions in cost and power consumption. It would be possible to build a fast memory subsystem through the use of multiple memory devices in a technique called bank interleaving. However, bank interleaving requires multiple memories and external logic, which significantly add to cost compared to a single Page Mode memory device. Page Mode devices can also reduce power in two ways. First, by reducing the time power must be at the higher active read consumption level. Second, in some Page Mode devices, by reducing the active read current requirement following the initial access time, such that the average power for a series of page reads is lower.

To demonstrate the speed advantage of a Page Mode device, we will compare the timing of the Am29PL160C to a standard flash device.

### HOW DOES A PAGE MODE READ WORK?

The first read from the page mode device is identical to a read from a standard flash device: CE# and OE#



**Figure 1. Timing Diagram For Am29PL160C Page Mode Read (Byte Mode)**

The Am29PL160C is available with  $t_{ACC}$  varying from 60 ns to 120 ns, but we will look at one in which  $t_{ACC} = 90$  ns, and  $t_{PACC} = 30$  ns. Figure 1 clearly depicts that the initial access time of 90 ns is required to do the first

read from a given page, but each subsequent read from the page is only 30 ns. Each read from within the page is triggered by changing A–1 to A2.

### Example: Comparing A Page Mode Device To A Standard Device

By comparing a page mode device and a standard device, the true advantage in speed can clearly be seen. Let's say that System A uses a Page mode flash memory device, and System B uses a comparable standard

flash memory device. Both systems are using the same processor, and both flash devices have the same random access time. Table 3 shows how much time is required by both systems to read eight consecutive bytes from memory.

**Table 3. Chart Comparing Read Times Between Page and Standard Devices**

	<b>System A: Am29PL160C</b>	<b>System B: Comparable Standard Device</b>
Byte 0	90 ns ( $t_{ACC}$ )	90 ns ( $t_{ACC}$ )
Byte 1	30 ns ( $t_{PACC}$ )	90 ns
Byte 2	30 ns	90 ns
Byte 3	30 ns	90 ns
Byte 4	30 ns	90 ns
Byte 5	30 ns	90 ns
Byte 6	30 ns	90 ns
Byte 7	30 ns	90 ns
<b>Total time</b>	<b>300 ns</b>	<b>720 ns</b>

Both systems require a  $t_{ACC}$  of 90 ns to do the first random access read. However, System A can now do subsequent reads in 30 ns, whereas System B still requires 90 ns to complete each and every read command. By adding all the access times together, you can see that System A, using a Page mode memory device, can read almost 2.5 times as fast as System B.

### Is The Speed Improvement Always Constant?

Although the previous example shows that a Page mode memory device can be dramatically faster than a standard flash, the improvement is not always constant. The more the system reads from the same page, the faster the average access time. The less the system reads from the same page, the slower the average access time. In order to obtain optimum performance from a Page mode device, the system's program should be structured such that as many consecutive reads as possible are from within the same page. This could involve aligning frequently accessed data structures or loop and jump labels to start on page boundaries.

### What About Program and Erase Operations?

Program commands work on one word/byte, and Erase commands work on one sector at a time, just as they do in standard Flash memories.

### What Are The Requirements On A System Using A Page Mode Flash?

The system must provide memory interface logic that is aware of the page address boundaries and the difference in access time between an initial access and a subsequent access within the same page so that the access time (number of wait states) can be adjusted dynamically.

Fortunately, many microprocessors already have such interface logic already integrated into their memory interface. Table 4 describes some of these processors.

**Table 4. Processor Families And Examples From Each Family**

<b>Processor family</b>	<b>Examples</b>
Motorola PowerPC	MPC850
Motorola Coldfire	MCF5307, MCF5206e, MCF5206
Sharp ARM	LH77790
Hitachi SuperH RISC	SH7709 (SH-3)

### Software Implications

In order to take full advantage of the performance enhancement of the Page Mode feature, initial accesses need to be minimized and page mode accesses need to be maximized, making the average access time as

low as possible. Ideally, in code, all branch target locations would be aligned at the beginning of pages, branch instructions would be located at the end of pages, with all the locations in between used by sequentially executed instructions. Ideally, all data structures would be aligned on page boundaries.

However, few if any code compilers have been optimized to the point to align all branch targets on programmer defined boundaries. Some optimizing compilers support specific techniques like instruction reordering and loop unrolling which will produce longer sequences of sequential instructions which in turn reduce code branching, but at the cost of lower code density. Code linkers can also start code modules at specific addresses or boundaries to force alignment. Extensive optimization of branch target locations would require either manual code optimization or some kind of code post-processor algorithm that would reorder instructions or insert No-Ops and recalculate the relative branch addresses in order to force branch target alignment.

Data structure alignment is easier to control since several compilers do support that function. One example is

the Microsoft compiler that comes with Visual Studio 6.0. One of the compile options is `/Zp[n]`, or “Struct Member Alignment,” where  $n$  is in bytes and can be 1, 2, 4, 8, or 16. Depending on what  $n$  is set to, the compiler will align data structures to the given page boundary. For example, `/Zp16` would put structures on 16-byte, or 8-word boundaries. However, this may lead to an increase in the memory space required as unused space may have to be inserted between each data record in order to align the start of each record.

## SUMMARY

Page Mode memory devices have been developed to increase system performance in spite of the market demands for higher density and lower voltage, that would otherwise tend to reduce memory performance. By changing the internal architecture, the Page Mode memory allows for faster read access times within each page. Using a Page Mode memory device can improve performance while holding or reducing cost and power consumption. But, the system must use the Page Mode feature properly in order to fully take advantage of the available performance advantage.

### Trademarks

Copyright © 2000 Advanced Micro Devices, Inc. All rights reserved.

AMD, the AMD logo, and combinations thereof are registered trademarks of Advanced Micro Devices, Inc.

Expressflash is a trademark of Advanced Micro Devices, Inc.