

# NAND Flash Performance Improvement Using Internal Data Move

## Introduction

Micron offers a NAND Flash feature known as internal data move (IDM). Memory devices equipped with IDM provide more sophisticated data management capabilities and higher system performance when designed in to audio, video, and mobile applications. In these applications, NAND Flash data is typically moved within the NAND Flash array for block management purposes.

In traditional NAND Flash devices, moving a block of data from one memory location to another requires the data to be read from the device externally, page by page. Then each page of NAND Flash data is post-processed for error correction. The data is finally reprogrammed to the new (and presumably erased) memory location in the NAND Flash array. This process is time-consuming, precludes other functions in the memory subsystem, uses excessive computing power, and slows system performance.

The Micron<sup>®</sup> IDM feature provides an alternative to this traditional solution in that IDM eliminates the extra steps required for external data movement. IDM makes it possible to move the same page of data internally, without any external read or write processes. Moving data internally saves time, reduces power consumption, and increases performance.

This technical note provides instructions for using the IDM feature in Micron NAND Flash memory.

## IDM Overview

IDM consists of an internal READ operation followed by an internal PROGRAM operation. This internal read/program process moves one page of data from its original memory location to another location in the NAND Flash array.

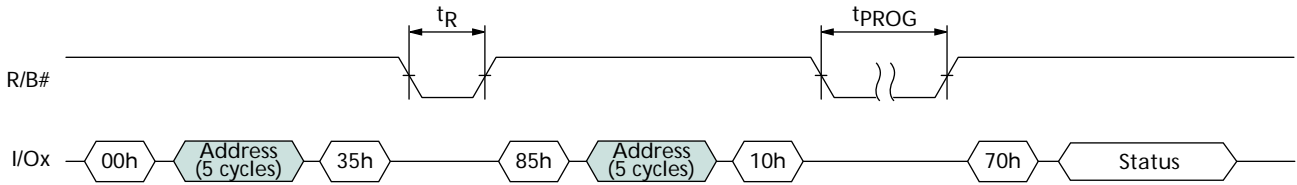
The internal READ operation is executed by issuing the READ FOR INTERNAL DATA MOVE (00h–35h) command and the source page address. Following the command, the device moves the addressed page of data from the NAND Flash to the cache register.

The cache register holds the addressed page of data until the PROGRAM FOR INTERNAL DATA MOVE (85h–10h) command is issued or the original command is aborted. If the command is aborted, data in the cache register is no longer valid.

The internal PROGRAM operation is executed by issuing the PROGRAM FOR INTERNAL DATA MOVE (85h–10h) command.

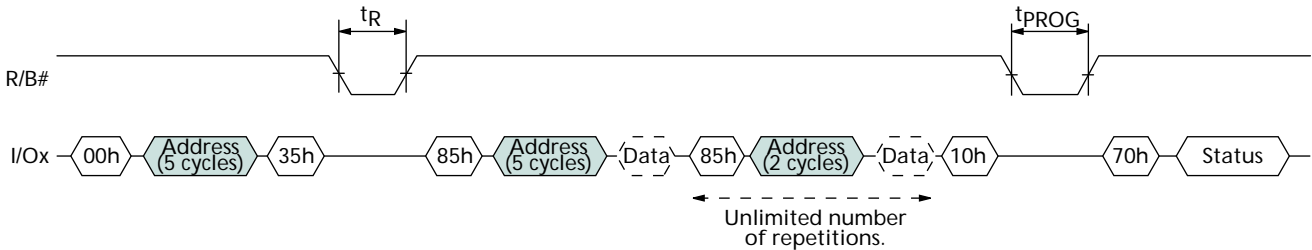
When the 10h portion of the PROGRAM FOR INTERNAL DATA MOVE command is issued, the device moves the page of NAND Flash data from the cache register to the new memory location in the NAND Flash array (see Figure 1 on page 2).

Figure 1: Internal Data Move



The 85h command can be used to reprogram the page of NAND Flash data in the cache register. As long as the page of NAND Flash data is held in the cache register, subsequent 85h commands with appropriate column addressing can be issued. This method enables reprogramming any portion of a page before issuing the 10h command to move the data to a new location in the NAND Flash array (see Figure 2). For a detailed explanation of this operation, refer to the appropriate Micron NAND Flash memory data sheet ([www.micron.com/nand/](http://www.micron.com/nand/)).

Figure 2: Internal Data Move with Random Data Input



## IDM Advantage

No external data operations occur as part of the IDM operation, so using IDM to move a page of NAND Flash data is considerably faster than using an external data operation.

The time required to move a block of NAND Flash data using traditional methods is 29ms (as shown in Table 1), compared to 20.8ms using Micron's IDM (as shown in Table 2). This equates to an 8.1ms (30 percent) time savings per block using IDM.

**Table 1: Traditional READ/PROGRAM Performance (x8)**

Mode	Activity	Parameter	Timing Unit	Repetitions	Standard PAGE READ Time	Unit	
READ/ PROGRAM	Command latch (00h)	<sup>t</sup> WC (MIN)	30	1	30	ns	
	Address latch	<sup>t</sup> WC (MIN)	30	5	150	ns	
	Command latch (30h)	<sup>t</sup> WC (MIN)	30	1	30	ns	
	R/B# LOW	<sup>t</sup> R (MAX)	25	1	25	µs	
	READ cycle	<sup>t</sup> WC (MIN)	30	2,112	63.4	µs	
WRITE	Command latch (80h)	<sup>t</sup> WC (MIN)	30	1	30	ns	
	Address latch	<sup>t</sup> WC (MIN)	30	5	150	ns	
	Program data cycles	<sup>t</sup> WC (MIN)	30	2,112	63.4	µs	
	Command latch (10h)	<sup>t</sup> WC (MIN)	30	1	30	ns	
	Program page	<sup>t</sup> PROG (TYP)	300	1	300	µs	
					Time required to read a page externally	88.6	µs
					Time required to read a block externally	5.7	ms
					Time required to program a page externally	363.6	µs
					Time required to program a block externally	23.3	ms
					Time required to move a page externally	452.2	µs
					Time required to move a block externally	29.0	ms

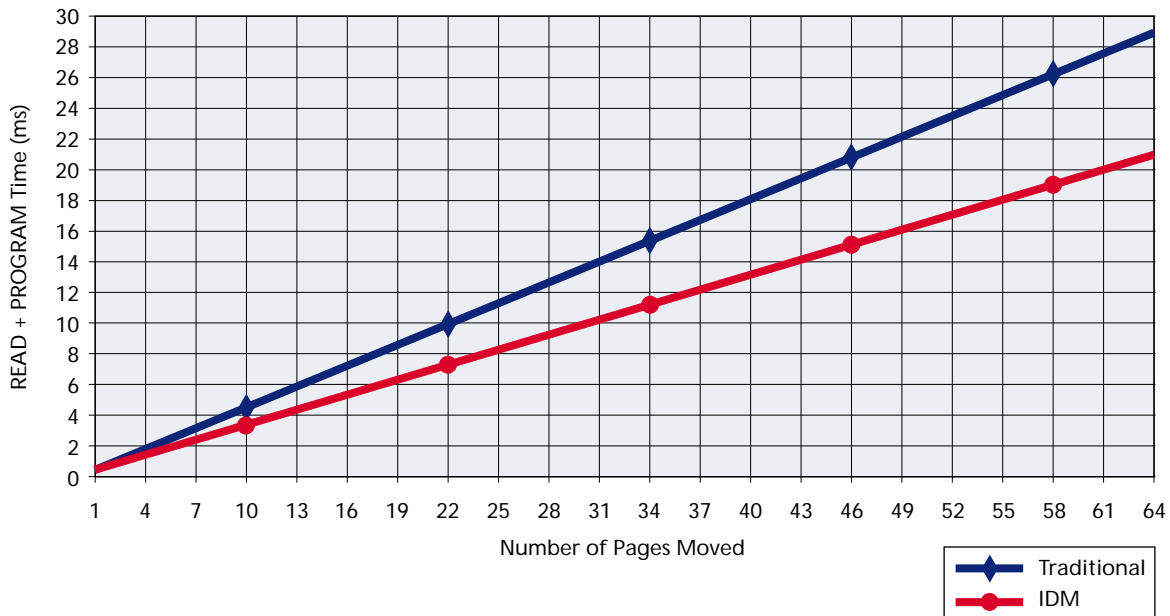
**Table 2: Internal Data Move with 512-Byte PROGRAM (x8)**

Activity	Parameter	Timing Unit	Repetitions	Total Time	Unit	
Command latch (00h)	<sup>t</sup> WC (MIN)	30	1	30	ns	
Address latch	<sup>t</sup> WC (MIN)	30	5	150	ns	
Command latch (35h)	<sup>t</sup> WC (MIN)	30	1	30	ns	
R/B# LOW	<sup>t</sup> R (MAX)	25	1	25	µs	
Command latch 85h)	<sup>t</sup> WC (MIN)	30	1	30	ns	
Address latch	<sup>t</sup> WC (MIN)	30	5	150	ns	
Data latch	<sup>t</sup> WC (MIN)	30	1	30	ns	
Program page	<sup>t</sup> PROG (TYP)	300	1	300	µs	
				Time required to move a page internally	325.4	µs
				Time required to move a block internally	20.8	ms

IDM gives designers the throughput they need to manage the volume of data required for today's audio, video, and mobile applications. With the ability to quickly move data from one memory location to another, designers can focus their efforts on creating more efficient and competitive systems.

As more data pages are moved using IDM, the time saved increases compared to traditional read/write data moves (see Figure 3 on page 4).

Figure 3: Internal Data Move Speed Comparison by Mode



## IDM Considerations

When data is moved internally using the IDM feature, there is no opportunity to perform error correction. As a result, excessive IDM operations without periodic checks may cause errors to accumulate, which could potentially decrease system reliability.

Data integrity depends on many factors, and no universal data integrity checks are available to prevent the effects of excessive IDM operations. Micron recommends using as many software techniques as possible to help minimize the risk associated with accumulating errors.

The most obvious approach for improving data integrity is to use a robust, multibit error-correction algorithm. In any error correction system, the higher the number of error correcting bits, the better. To achieve the highest data integrity when using IDM, designers should consider these factors:

- Error correction computing capabilities
- Maximum number of anticipated IDM operations
- Required level of system reliability

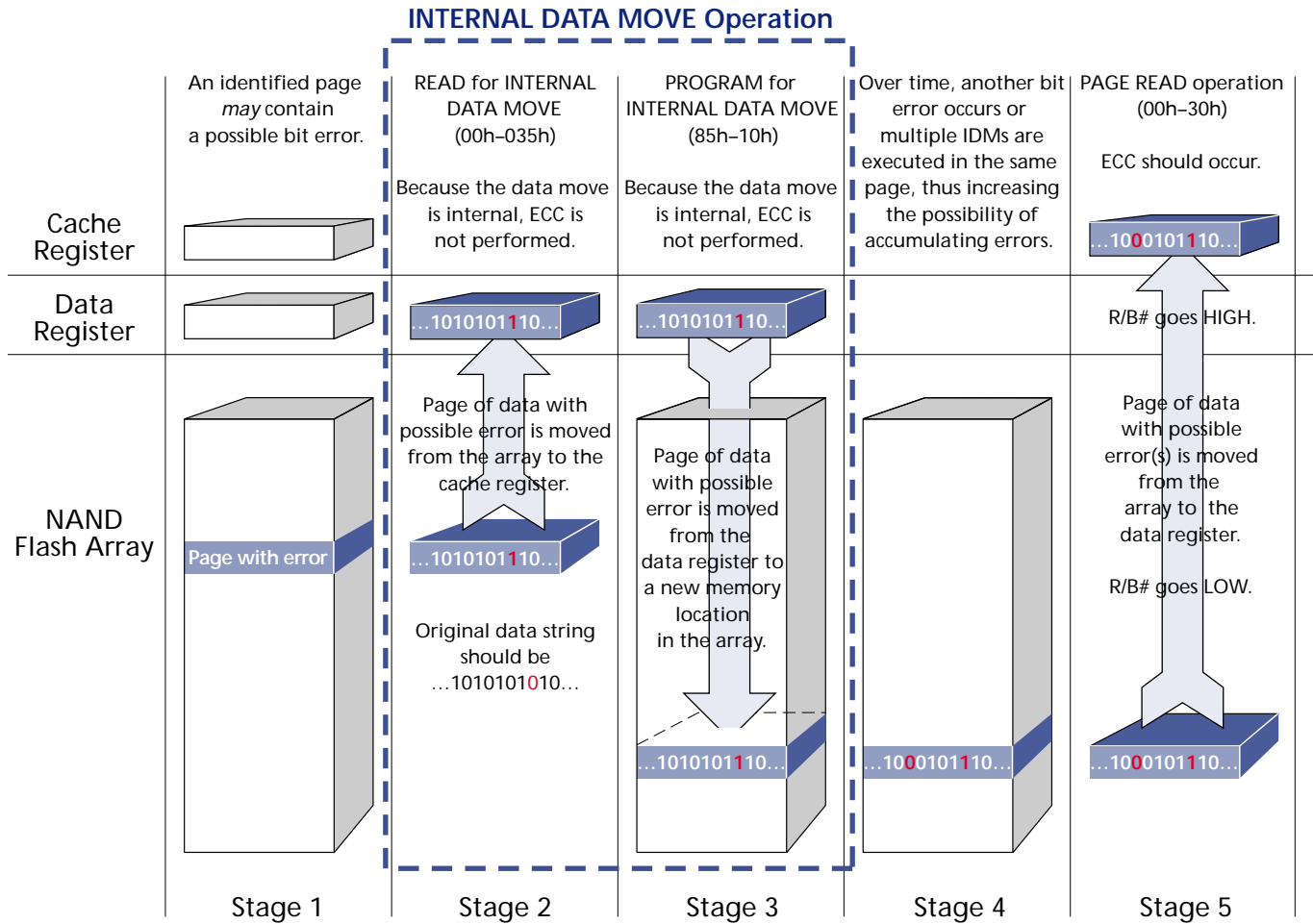
Designers can also check the integrity by performing a post-READ operation following an IDM operation. The optimum frequency for post-READ operations depends on the desired level of error correction and the number of IDM operations implemented between postchecks.

Combining a multibit error-correction algorithm with a post-READ operation following IDM operations should provide a high level of data integrity. Designers can also choose to mark a block “bad” if more than two or three bit errors occur in any single block. Again, data integrity depends on many factors, and no universal data integrity guidelines or rules guarantee an error-free system.

## IDM Error Scenario

Figure 4 depicts an IDM error scenario. The example shows the five necessary steps in an IDM operation.

Figure 4: INTERNAL DATA MOVE Operation



## Stage Descriptions

The stages listed refer to the IDM error scenario presented in Figure 4.

### Stage 1

A NAND Flash array exists with an error in a page. The page was originally programmed to contain the data string 1010101010. However, normal NAND Flash operation may cause data corruption. Now the data string shows 1010101110, with an error in bit location 2.

### Stage 2

This page of data is moved internally from the array to the cache register, using the READ FOR INTERNAL DATA MOVE (00h–35h) command. The error tracks with the IDM operation.

### Stage 3

Because the data are moved internally or reprogrammed using the IDM (85h–10h) command, no error correction occurs. The undetected and uncorrected error follows the move operation to the new memory location.

### Stage 4

Over time, normal NAND Flash operations without error correction may result in another bit error. The data string ends up as 100010110, with errors in bit locations 6 and 2.

### Stage 5

The page of data is finally read from the NAND Flash device. However, the number of accumulated errors exceeds the repair capability of conventional error-correction techniques, so the data is corrupted.

## Conclusion

The Micron NAND Flash IDM feature offers designers up to a 30 percent speed advantage over traditional NAND Flash READ/PROGRAM data-move operations. Individual system configurations, adequate error correction capability, and the number of IDM operations used will all factor in to final performance results. When precautions are in place to control excessive internal data moves, the accumulation of unresolved errors that may cause data corruption can be minimized. Designers should implement system-level techniques as discussed in this technical note to ensure data integrity and the highest possible performance and system reliability.



8000 S. Federal Way, P.O. Box 6, Boise, ID 83707-0006, Tel: 208-368-3900  
prodmktg@micron.com www.micron.com Customer Comment Line: 800-932-4992  
Micron, the M logo, and the Micron logo are trademarks of Micron Technology, Inc.  
All other trademarks are the property of their respective owners.

## Revision History

<b>Rev. B</b> .....	<b>6/07</b>
<ul style="list-style-type: none"><li>• Figure 4 on page 5: Updated left labels from I/O buffer to cache register, and cache register to data register.</li><li>• “Stage Descriptions” on page 6: Underlined and made the changed numbers in the data strings red for visibility.</li></ul>	
<b>Rev. A</b> .....	<b>9/06</b>
<ul style="list-style-type: none"><li>• Initial release.</li></ul>	