Using Erase Suspend and Erase Resume Functions in NOR Flash

Introduction

The Erase Suspend function can be used to temporarily stop an Erase operation to allow for the reading or programming of other blocks in the same Flash memory. The optimal implementation of Erase Suspend in the system is described in this document. The descriptions included in this document are general and not related to any particular Macronix Flash memory family. Therefore, please refer to the device datasheet for details about the commands, timing and limitations.

Erase and Erase Suspend/Resume

NOR Flash memory requires that a previously written area be erased before it can be rewritten. Erase sizes vary from 4KB to 256KB, depending on the selected Macronix part. In serial flash, Macronix typically defines a block as 64KB or 32KB, which are further divided into 4KB sectors. In parallel flash, the terms “blocks” and “sectors” are used interchangeably. Typically, a single 128KB sector Erase can take 600ms to complete. While an Erase operation is progress, no other operations such as Read or Program can be performed.

In “Real Time” systems, sometimes it is necessary to execute critical tasks that involve reading or programming while the Erase operation is in progress. Long Erase time latency is not tolerable in such cases. The effect of the Long Erase latency can be reduced by using the Erase Suspend and Erase Resume functions. The Erase Suspend command can pause an Erase operation to allow the system to read or to write the memory. As soon as the critical read or write has completed, the Erase Resume command can be used to continue the interrupted Erase operation, as shown in Figure 1.

Figure 1. Read while Erase is Suspended

![Diagram showing read while erase is suspended](image-url)
Erase Operation Details

The Erase operation sets a memory sector or block to the all “1’s” state. The Erase operation requires a proper sequence of phases to succeed. The Erase algorithm is automatically managed by an internal State Machine, which controls a looping sequence of Erase pulses and Verify steps as shown in Figure 2.

First, all the bytes in the sector being erased are written to all “0’s” (this initial step is not shown in Figure 2). To set the memory cells to the ‘1’ state, erase pulses with a high electric field are applied to the sector to remove electrons from the floating gate of every memory cell in the sector/block. Then the Verify operation sequentially reads off all of the address locations in the sector/block to check whether they have all been erased to ‘1’. When the entire sector is verified to be ‘1’ then the Erase operation is complete.

The Verify operation starts from the first address of the sector/block being erased. If the data at that address is verified to be erased (all “1’s”), then the next physical address is verified. The Verify operation continues until a single Verify fails or until the last address of the sector/block has been reached. If the result of a Verify is “fail” then a new Erase pulse is applied before continuing with the Verify operation, starting from the last failing address.

Figure 2. Erase Algorithm
Figure 3 shows a timing diagram of the Erase sequence internal to the Flash memory. Early in the sequence, the Verify operation quickly detects “0’s” and triggers a new Erase Pulse because very few memory locations have been erased. Gradually the Verify operation runs longer as more locations within the sector/block verify successfully until the erase completes.

Figure 3. Erase Flow: Erase/Verify Sequence

An Erase Suspend request can occur during an Erase Pulse or during a Verify operation in an unpredictable manner. If the suspend request occurs during an Erase pulse, as shown in 4, the Erase pulse is interrupted because it would be too long to wait for the Erase pulse to complete. When the Erase pulse is interrupted, the erase efficiency is reduced. If the truncated Erase pulse is much shorter than the standard erase pulse duration, its erasing effect can be negligible. This not a problem because a new Erase pulse will be started when the Erase is resumed. However, in the case of multiple Suspend/Resume cycles, if the resume to suspend time is much smaller than the standard Erase pulse, a degradation of the total erase time might be observed.

Figure 4. Erase Flow: Erase/Verify Sequence with an Erase Suspend Event.
Erase Suspend and Erase Resume Commands and Timing

After receiving a Suspend command, the device takes a maximum of 20us to enter the Erase-Suspended mode as shown in Figure 5. After this latency has elapsed it is possible to proceed with a Read Command.

Suspend latency is designed to be as short as possible in order to allow the system to read data from another sector quickly. Erase Suspend status can be recognized by polling the proper bit in the Status or Configuration register. For a complete description of the flow please refer to the specific flash datasheet.

If the Erase operation is suspended, other sectors/blocks (other than the one suspended) can be read. While Erase is suspended in a sector/block, there might be some limitations to read from a certain group of sectors/blocks. This possible limitation should be confirmed in the specific flash datasheet of the device being used.

Figure 5. Erase Suspend Command and Suspend Latency

The Erase Resume command will resume a suspended Erase as shown in Figure 6. \( T_{\text{erase}} \) is the time it takes for the Erase operation to finish after being resumed. A Read Command is permitted after the Erase operation completes.

Figure 6. Erase Resume Command

It is permitted to suspend the same erase operation multiple times but a typical Resume-to-Suspend time should be observed in order to avoid excessive delays to the completion of the erase operation.

Resume-to-Suspend time is also referred to as Resume Latency or \( T_{\text{resume}} \) in this document. Resume latency is shown in Figure 7.
A typical value for $T_{\text{resume}}$ is between 400\,us and 1\,ms depending on the device and technology. Refer to your Macronix datasheet to get the actual number for the product of interest.

Datasheets usually indicates only a typical value for $T_{\text{resume}}$ instead of a strict minimum requirement. The reason is to provide flexibility in setting the appropriate resume latency for the application. In fact, 1\,ms $T_{\text{resume}}$ latency is quite long, especially if compared to the initial Suspend latency of 20\,us. Some critical systems may require, on special conditions (e.g. critical interrupts), to react in the shortest time and cannot afford to wait for 1\,ms before accessing the memory again.

In some Flash memories (e.g. MX25U family) a minimum $T_{\text{resume}}$ is defined. Using such a timing will allow the system to reap the benefit of a very short suspend time in any situation, but the user is warned that repeatedly using such a minimum timing will prevent the Erase from progressing.

In the following section it will be described how to set and use the correct $T_{\text{resume}}$ latency in the application and to avoid deadlock situations.

**Erase Suspend/Resume Flow**

System developers are allowed to shorten the typical resume latency $T_{\text{resume}}$, but they need to be aware that the overall erase time might become longer. If a shorter than typical $T_{\text{resume}}$ is used, designers need to ensure that the system doesn’t enter into a critical Resume-Suspend loop. In this situation the Erase could take an excessively long time to complete or even never complete, if a very short $T_{\text{resume}}$ is set.

In order to properly design the Erase Suspend function it is important that the system designer understand:

i. What is the acceptable resume latency for the application

ii. What situations will trigger a suspend request, and what is the probability of it occurring

If it is not critical to use the absolute minimum suspend latency, designers are recommended to use the typical (sometimes referred to as “standard”) suggested resume latency datasheet timing. Because the issuance of a Suspend Request is an asynchronous event, in order to guarantee $T_{\text{resume}}$ timing, it is necessary to implement either a fixed delay or a timer in the flow. See examples in Figure 9 and Figure 10 in the Appendix for reference.

In cases where system timing is critical and lowering the resume latency to very short values is required (e.g. down to 1\,us when allowed as for MX25U family), then it is important to understand if the software could fall into an endless loop because of Erase Time enlargement. The system designers that need to use aggressive $T_{\text{resume}}$ timing must adopt some countermeasures to avoid this situation.

If a Suspend request occurs only once or very infrequently (like in the case of a power loss) then resume latency can be reduced to minimum values without concern. Once the critical interrupt has been serviced, the system will return to standard operation and will have a chance for the Erase operation to progress.

If the user is uncertain about the suspend/resume frequency, an easy solution to ensure erase completion is

![Resume to Suspend Time Diagram](image-url)
to insert a fixed ‘typical’ $T_{\text{resume}}$ delay after a fixed number of suspend commands have been executed (e.g. 10).

It is important to remember that if a shorter than typical $T_{\text{resume}}$ is required, the user can characterize the behavior of the memory under such conditions to find out the Erase time enlargement when Resume/Suspend is issued repeatedly under such timing condition.

It is important to highlight that when using a $T_{\text{resume}}$ below the recommended value the total Erase time is no longer guaranteed.

**Limitations of Erase Suspend**

There is no limit regarding the number of times the Suspend/Resume commands can be issued. However, complete erase pulses are required for an erase operation to progress towards completion as explained in previous sections.

In the Macronix GL family (parallel NOR flash) there is a limitation regarding the sectors (forming a group) that can be read during Erase Suspend. Please refer to the datasheets or Application Notes to be aware of such limitations.

**Erase Suspend and Simultaneous Read/Write**

Multibank products can offer a Simultaneous Read/Write feature (sometimes called Read While Write). The memory in Multibank Products are typically partitioned into two, four or eight banks depending on the memory size. The Simultaneous Read/Write feature allows reading from one bank of the memory while an Erase or Program operation is on going in another bank as shown in Figure 8.

![Figure 8. a) Erase Suspend versus b) Simultaneous Read/Write Operation](image)

Multibank products offer great flexibility to the systems but at a price of a greater cost because read and service circuits are duplicated in each bank. Multibank devices can be used when there is the need for updating some data while an unstoppable process is executed. The use of Multibank devices is predominantly found in mobile applications. Macronix offers Multibank products for 1.8V applications.

In many applications, efficient Erase Suspend/Resume features makes Read-While-Write operation not strictly necessary. In fact, if the response to suspend commands is short enough, then the system can still work without using the Simultaneous Read/Write feature. Macronix products are designed to offer optimal Suspend/Resume performance.
Summary

With a better understanding of how the Macronix Erase Suspend/Resume function works, the feature can be implemented in such a way as to enhance system performance.

References

<table>
<thead>
<tr>
<th>Macronix Doc.</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AN0202</td>
<td>Design notice - Serial Flash Erase Suspend/Resume &amp; Program Suspend/Resume Flow</td>
</tr>
</tbody>
</table>

Revision History

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>01</td>
<td>May 3, 2013</td>
<td>Initial release</td>
</tr>
</tbody>
</table>
Appendix: Erase Suspend/Resume Flow Examples

Figure 9. Erase Suspend/Resume Flow with Tresume Timer

Start

Tresume Timer = 0

Erase Command

RDSR Command

WIP = 0

Erase Suspend Request

Y

N

Tresume Timer = 0

Y

N

Erase Suspend Command

Read Data from other sectors

Erase Resume Command

Set Tresume = Tresume_typ

End
Figure 10. Erase Suspend/Resume Flow with Tresume Delay

Start

Erase Command

RDSR Command

WIP=0

Y

End

N

Erase Suspend Request

Y

N

Erase Suspend Command

Read Data from other sectors

Erase Resume Command

Wait Tresume_typ
Except for customized products which have been expressly identified in the applicable agreement, Macronix's products are designed, developed, and/or manufactured for ordinary business, industrial, personal, and/or household applications only, and not for use in any applications which may, directly or indirectly, cause death, personal injury, or severe property damages. In the event Macronix products are used in contradicted to their target usage above, the buyer shall take any and all actions to ensure said Macronix's product qualified for its actual use in accordance with the applicable laws and regulations; and Macronix as well as it's suppliers and/or distributors shall be released from any and all liability arisen therefrom.

Copyright© Macronix International Co., Ltd. 2012. All rights reserved, including the trademarks and trade name thereof, such as Macronix, MXIC, MXIC Logo, MX Logo, Integrated Solutions Provider, NBit, NBiit, Macronix NBit, eLiteFlash, XtraROM, Phines, KH Logo, BE-SONOS, KSMC, Kingtech, MXSMIO, Macronix vEE, Macronix MAP, Rich Audio, Rich Book, Rich TV, and FitCAM. The names and brands of other companies are for identification purposes only and may be claimed as the property of the respective companies.

For the contact and order information, please visit Macronix's Web site at: http://www.macronix.com