NOR Flash Memory Erase Operation

The purpose of this document is to provide information about the internal processes and algorithms occurring during an Erase operation. This information is at a level that allows the user/systems designer to properly design a robust and functional system. It also discusses proper procedures to follow for an orderly execution and completion of these operations, as well as how to avoid getting into situations that might affect data integrity.

Contents

1. Introduction

In today's technology-driven world, gadgets, mobile devices, and other electronic equipment rely on NOR Flash memory to store code for execution; for storing important system parameters, calibration data; for data logging; or other applications requiring fast non-volatile memory.

One of the fundamental principles of the NOR Flash memory is that it must be erased before it can be programmed. Another important characteristic is that the erase operation must happen over an entire block of memory simultaneously (in bulk), rather than sequentially in a byte-by-byte fashion.

The above two characteristics dictate an implementation requiring the Erase operation to be separated into three distinct phases: Pre-Program, Erase, and Recovery.

To enable robust and reliable operation, it is important for the system designer to understand these underlying processes, as well as the potential undesirable side-effects if any of the three phases are asynchronously interrupted by intentional or accidental events (for example: programmatic erase-suspend operation, power supply brown-out, or glitch).

The descriptions included in this document are general and not specifically targeting a particular Renesas Flash device.

The design, technology, algorithms, and implementation details are the sole property of Renesas, protected by patents and governed by IP (Intellectual Property) laws, and, therefore, purposefully not discussed in this document.

The diagrams and pictures are conceptual in nature.

For details about the commands, timing, and limitations. see the device datasheet.

2. Storing Information in a Flash Memory

The basic storage element (cell) of a memory is comprised of a MOSFET with a “floating gate.” The amount of electrical charge trapped inside the floating gate of the memory cell defines whether the cell is **Programmed** or **Erased**.

Figure 1 shows the physical composition and the symbol of a floating gate MOSFET.

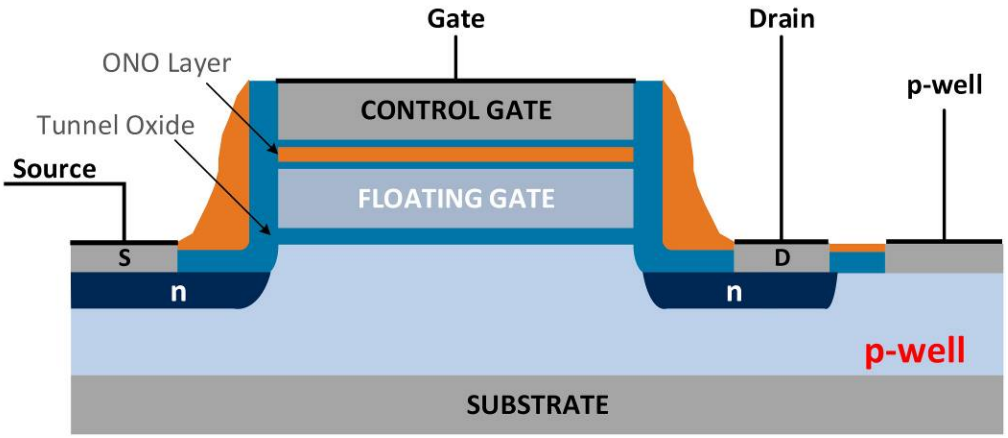

Figure 1 Symbol and Structure of a Floating Gate MOSFET

To store large amounts of information, the Flash memory is organized into arrays of memory cells. The memory array is a matrix of N rows by M columns. At each row-column intersection there is exactly one memory cell (a MOSFET with a floating gate). This MOSFET stores one bit of information.

The capacity of the memory array (in bits) is calculated as N [rows] x M [columns] (see Figure 2)

By convention, the rows are called Word-Lines (WL), and the columns Bit-Lines (BL).

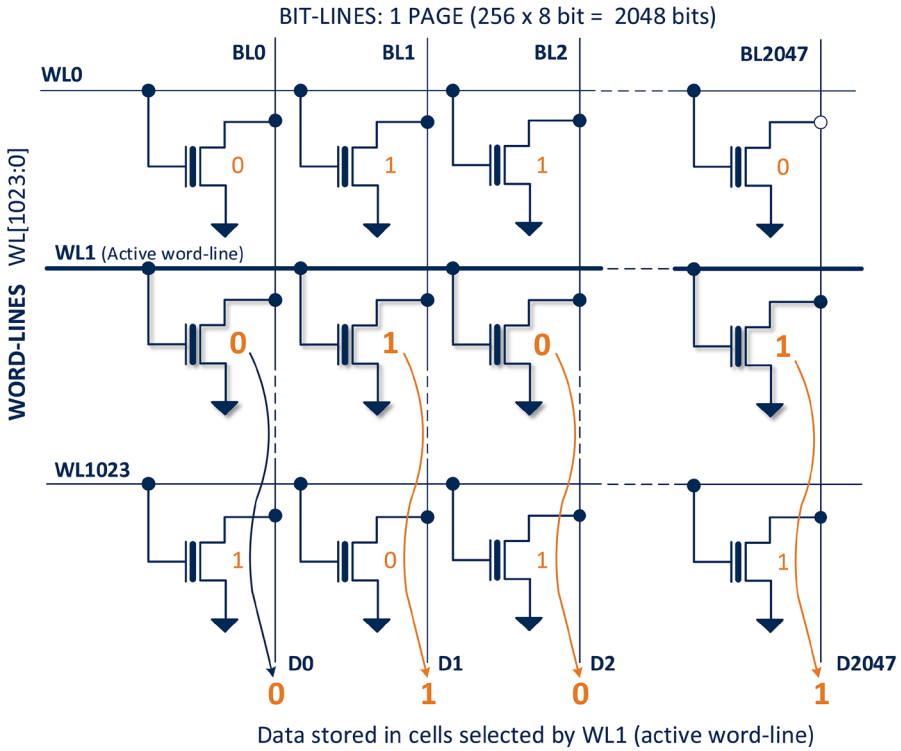

Figure 2 Organization of the Memory Array

A Page is a collection of Bit-Lines attached on a single Word-Line. One Word-Line can contain one or more Pages. One Page typically consists of 256 or 512 bytes of memory (2048 or 4096 bits).

Silicon fabrication requires that all memory cells be built upon a common foundation, called a **substrate**.

To make memory operations more manageable, a range of p-wells are implanted into the substrate. These p-wells are electrically insulated from each other and are called **common p-wells** (see Figure 3). Arrays of memory cells are embedded in the common p-wells. Each p-well has its own electrode, allowing it to be biased with specific voltages under the control of the memory controller.

The collection of memory cells embedded into a single common p-well is referred to as a **Physical Block** or **Partition**. In this document, these two terms are used interchangeably.

The typical size of a physical block is 1 to 8 Mbit. This can vary from device to device, depending on device family, memory organization, and capacity. Typically, this information is not spelled out explicitly in the datasheet.

One key point to remember is that in each Physical Block the p-well and the Bit-Lines are shared. This has a profound effect on device operation under some circumstances, as described later in this document.

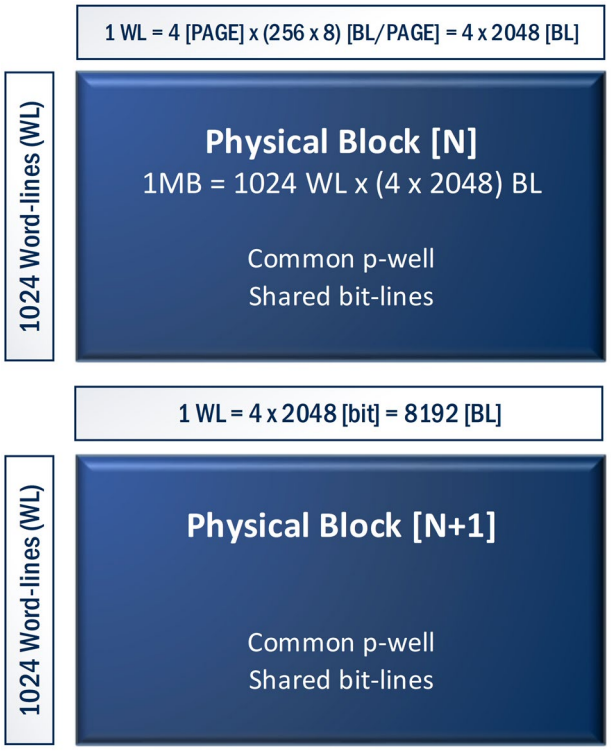

Figure 3 Organization of a Physical Block / Partition (With Common p-well and Shared Bit-Lines)

For convenience, the Physical Blocks are further divided into smaller **Logical Blocks**; this allows the user to work with smaller, more manageable memory sizes (see Figure 4).

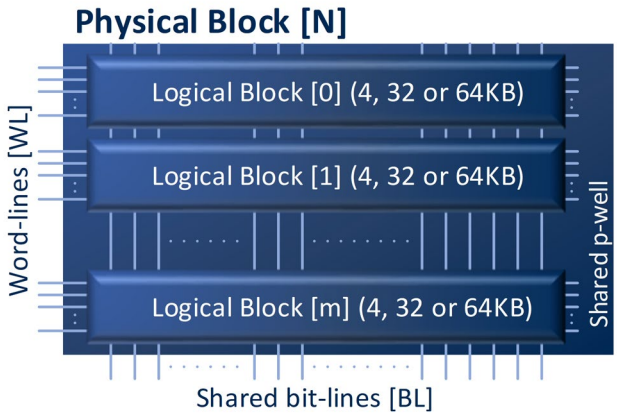

Figure 4 Detailed View of a Physical Block / Partition Containing a Range of Logical Blocks. Note the Shared p-well and Bit-Lines

The typical Erase block sizes are 4 kB, 32 kB and 64 kB. Some devices support even smaller memory block sizes, for example 256 bytes (size of a single Page). Additional benefits of using smaller memory blocks are that certain internal operations can be pipelined and parallelized, resulting in improved erase performance (for example: shorter erase times).

This type of partitioning is logical in nature. These blocks are still part of the larger Physical Blocks.

Figure 5 shows a picture of an actual 16 MB Renesas Flash memory device with sixteen Physical Blocks [15:0], 1 MB each.

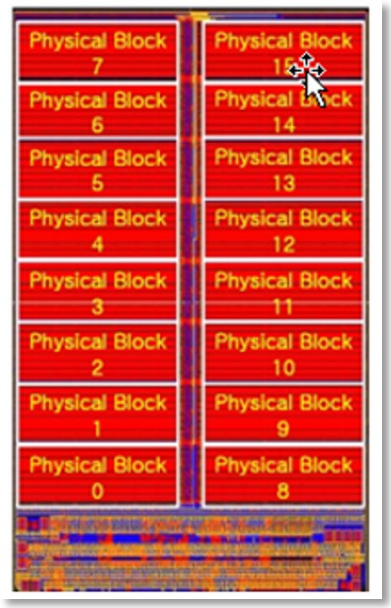

Figure 5 Organization of a 16 MByte Device with Sixteen 1 MByte Physical Blocks

3. Flash Memory Operations

There are three main operations that can be performed on any flash memory: **Read**, **Program**, and **Erase**.

- One of the key concepts inherent to NOR Flash technology is that it must be erased first, before a previously programmed memory block can be (re)programmed.
- Another important characteristic is that the Erase operation involves pre-programming of already erased cells.

While an operation is being executed, after the completion of each step, internal checks are performed to ensure the operation in progress is executing properly.

To successfully erase a NOR Flash, all three operations permissible on a NOR Flash (erase, program and read) are internally invoked as part of the Erase operation. For that reason, to fully understand Erase, we must also become familiar with Program and Read operations.

3.1 Read

To determine the state of a cell (whether it was programmed or erased), the content of the cell must be read. It is done by applying a reference voltage of approximately 5.5 V connected to the Gate of the MOSFET (V_{GS}). This voltage is called the “reference threshold voltage” and is denoted by V_{TREF}. In other words, the Gate-Source (V_{GS}) voltage is set to equal V_{TREF}, as shown in Figure 6.

If the MOSFET conducts current at that level, its state is defined to be logical ‘1’; otherwise, when the MOSFET does not conduct current, it is in a logical ‘0’ state.

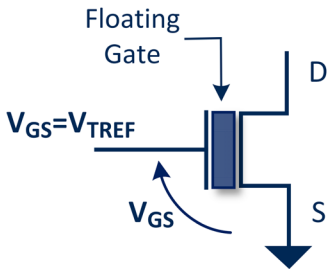

Figure 6 MOSFET Symbol and Associated Voltages During a Read Operation

The Gate-Source Threshold voltage (V_{GSTH}) determines whether a MOSFET conducts current:

- If the applied Gate-Source voltage (V_{GS}) is smaller than the Gate-Source Threshold voltage (V_{GSTH1}) ($V_{GS} < V_{GSTH}$), the MOSFET does not conduct current, and, by convention, it is defined to be in a logical ‘0’ state. In shorthand notation: **DOES NOT CONDUCT CURRENT → ‘0’**
- If the applied Gate-Source voltage (V_{GS}) is larger than the applied V_{GSTH} ($V_{GS} > V_{GSTH}$), the MOSFET conducts current, and it is defined to be in a logical ‘1’ state. **CONDUCTS CURRENT → ‘1’**

When determining the state of a cell, a reference Gate-Source voltage (also called V_{TREF}) of 5.5 V is applied to its Gate. Integrated Current Sense Amplifiers and Comparators measure the current levels flowing through the MOSFET, and, based on the measured current levels, the state of the cell is determined.

Note that for a specific cell, V_{GSTH} is not constant. The cell can be programmed and erased many times over the lifetime of the device. With each program and erase (depending on many factors, such as the length, strength, and number of Program/Erase pulses, power-supply voltage levels, the level of wear-and-tear of the cell and other factors), the number of trapped charges inside the floating gate also varies significantly. Consequently, V_{GSTH} also changes.

For example, consider a cell that is in Erased state. The floating gate of such cell is depleted of trapped electrons, hence its V_{GSTH} is relatively low (below 5.5 V). If a V_{TREF} = 5.5 V is applied, the MOSFET conducts current (blue dot in Figure 7); thus, it is in a logical '1' state.

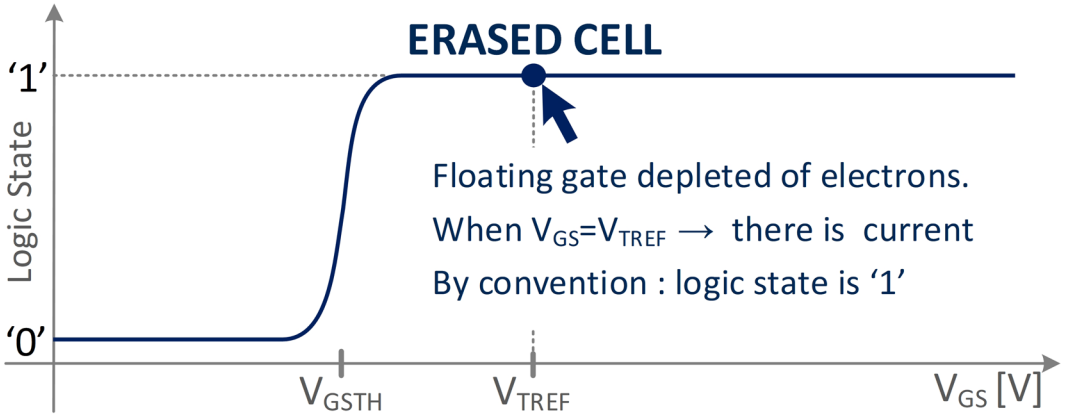

Figure 7 Cell in Erased State

When the cell is in a Programmed state, the floating gate has high concentrations of trapped electrons, so its Gate-Source Threshold (V_{GSTH}) voltage is higher than the applied V_{TREF} ($V_{GSTH} > V_{TREF}$).

To determine the state of this cell, a reference voltage V_{TREF} = 5.5 V is applied to its Gate. In this case the MOSFET does not conduct current (red dot in Figure 8); thus, it is in a logical '0' state:

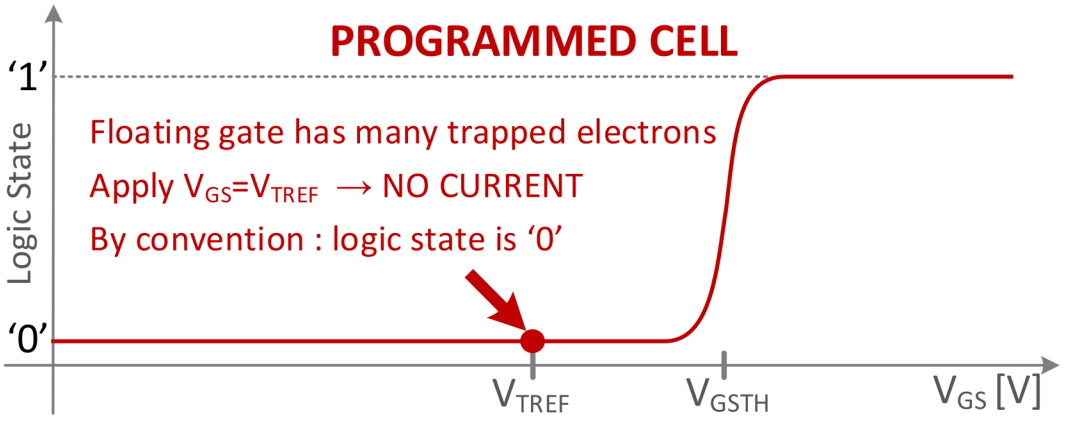

Figure 8 Cell in Programmed State

The read operation is also often performed as part of internal algorithms, as an intermediate step, to verify whether a cell was sufficiently programmed or erased (margins). In these cases, V_{TREF} is set to any value between 0 and the maximum allowable for that device, suitable for the objective of that operation.

The applied Gate-Source voltage (V_{GS}) in this context is referred to as V_T (instead of V_{TREF}). Among others, V_T is used to establish the so-called V_T curves, that are discussed later in this document.

As seen later in the document, when concepts of Pre-program and Erase are introduced, during Pre-program the V_T is set to a voltage level corresponding to "Pre-program Verify", and during an Erase operation it is set to "Erase Verify".

Based on the results of these intermediate checks, the algorithm keeps repeating the on-going operation (Erase or Program), until the required reference voltages are reached, or until a predetermined number of attempts has been reached. These concepts are discussed in more detail later in this document.

3.2 Program

The Program operation is based on the physical process called “Hot Electron Injection”. The Gate of the MOSFET is connected to a large positive voltage (+9 V). Simultaneously the Source is positively biased to about +4 V, causing electron flow from the Source to Drain. The electrons, on their way from Source to Drain, pass underneath the positively biased floating gate, as shown in Figure 9.

Due to the strong positive electrical field applied to the Gate, some of the electrons are “pulled into” the floating gate. Once inside, these electrons no longer have the required energy to escape the confines of the floating gate, and they remain trapped: the cell is programmed.

While the programming is in progress, the device is in busy state, as indicated by the RDY#/BSY bit set to 1 in the status register.

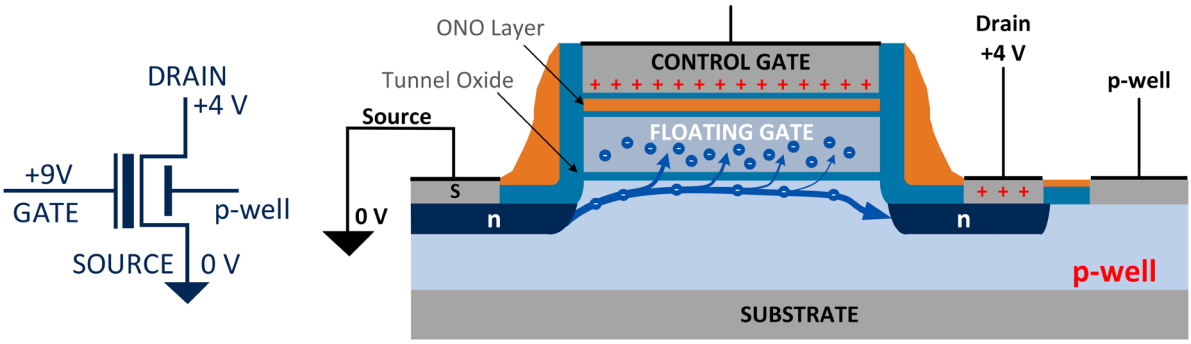

Figure 9 Programming a Memory Cell

The Gate-Source Threshold voltage (V_{GSTH}), required for the MOSFET to start conducting current is largely determined by the number of electrons trapped inside the floating gate (in other words: how “strongly” the MOSFET was programmed). The larger the concentration of trapped electrons inside the floating gate, the stronger the cell was programmed and the higher its Gate-Source threshold voltage.

For a cell to be placed into a valid programmed state, its threshold voltage must be larger than ~5.5 V. Ideally, there should be some margin, so the desired voltage should be at least 6.5 V.

3.3 Erase

The Erase operation is happening through a process known as “Fowler-Nordheim Tunneling.”

A high, positive voltage (approximately +8 V) is applied to the common p-well, attracting trapped charges inside the floating gate of the MOSFET. Simultaneously, high negative voltage (-10 V) is applied to the Gate of the MOSFET, that repels the trapped negative charges. The Source and Drain of the MOSFET are left unconnected (see Figure 10).

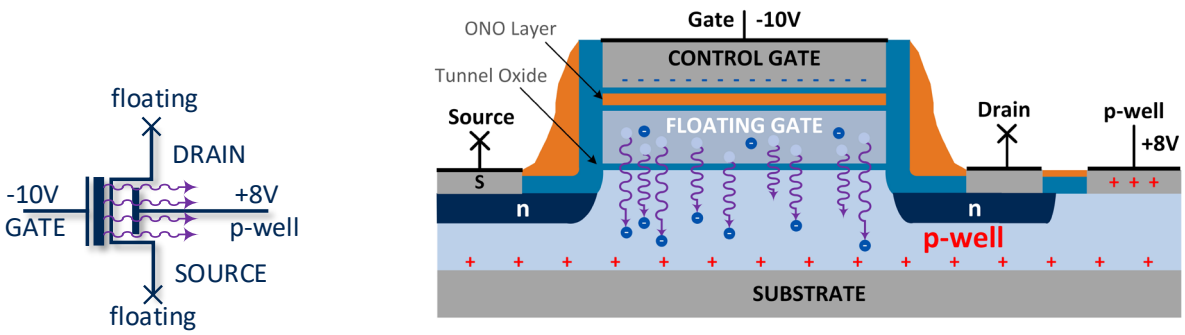

Figure 10 Erasing a Cell Through Fowler-Nordheim Tunneling

These two voltages superimpose and create a large electrical field that enables electrons to “tunnel” through the oxide barrier. Through this process, the floating gates are depleted of electrons, lowering their threshold voltage.

The currents are minute if looking at the individual cell level (typically in the nA range). Because of the very small currents, to extract sufficient number of electrons, the electrical fields must be maintained for a long time, making the Erase operation comparatively slow.

While the erase operation is in progress, the device is in BUSY state (RDY#/BSY is asserted in the Status Register).

3.4 Program vs Erase

Turning a '1' into a '0' is called a **Program** operation. It is always possible to individually program a cell.

Turning a '0' into a '1' is called an **Erase** operation.

The way the NOR Flash memory is structured, it is impossible to individually erase memory cells. Erase must happen in bulk: erasing entire logical blocks simultaneously.

When comparing the Erase and Program operations, we can observe a few important facts:

Program is a fast operation, requires more energy than erase, but is more concentrated in time – one byte or word at a time, which makes it a more controlled operation. Programming is typically performed iteratively, by applying programming pulses. Each programming pulse injects a certain number of electrons into the floating gate. At the end of each pulse, the threshold voltage is verified. If the desired level has not yet been reached, another programming pulse is applied. This process is repeated until the desired margins are achieved or until a specified number of attempts has been reached and the controller "gives up." For the devices that support this feature, an ERASE/PROGRAM ERROR (EPE) flag is asserted.

The erase operation is slow compared to program and read operations because the "extraction" of trapped electrons (tunneling) from the floating gate is a slow process, and also because it happens concurrently over an entire logical block of memory, where each cell starts from a slightly different level of trapped charges.

For a typical Flash device, it can take 60 ms, 200 ms, and 350 ms to erase a 4-kB, 32-kB, or 64-kB block, respectively. Just like the program, the erase operation is also iterative.

While the Program operation is faster than the Erase operation, (typically <5 µs for a byte), it also happens at a finer resolution (at individual byte or a size, which is typically 128 or 256 bytes).

IN SUMMARY:

In a NOR Flash memory, **Programmed** bits are stored as logical '0s,' and **Erased** bits as logical '1s'.

Operation	State Transition	Individual or Bulk Operation	Speed
Programming	'1' → '0'	Byte or Page level (1.to.256 Bytes)	Fast operation
Erasing	'0' → '1'	Bulk (typically 4, 32, 64 kB or chip)	Relatively slow

4. V_T Distribution

To better explain what is happening during a Block Erase operation, we must introduce the concept of the V_T distribution curve. This is a histogram that plots the number of cells as function of V_T. It is similar to Figure 7 and Figure 8, except it is plotted over the entire population of the memory array.

V_T is a Gate-Source voltage (V_{GS}) that is required for a MOSFET in the cell to conduct electrical current at a specific level (typically 10 µA or higher). It correlates with the number of trapped charges in the floating gates.

Cells with a lot of trapped electrons (programmed cells), have a high V_T, typically 6.5 V or higher.

Cells depleted of electrons (erased cells) have a low V_T, typically 4 V or lower.

Because during an erase multiple bits are erased in parallel, the distribution of V_T can be quite large. And it is possible for a memory cell to be put into an **Over-Erased** state. An over-erased cell conducts a small amount of current even when its Gate voltage is close to 0 V: this is called *leakage current*. Such a state is problematic for a NOR Flash architecture because of shared Bit-Lines, and it must be managed.

Figure 11 shows the V_T distribution of a 32 Mb (4 MB) NOR Flash device that was programmed with a checkerboard pattern, where half of the cells (16 Mb) are programmed, and the other half (16 Mb) erased.

The diagram consists of two distinct humps: one corresponds to Erased cells, peaking at around a V_T of 3 V. The other to Programmed cells. This one peaks at approximately V_T of 8 V.

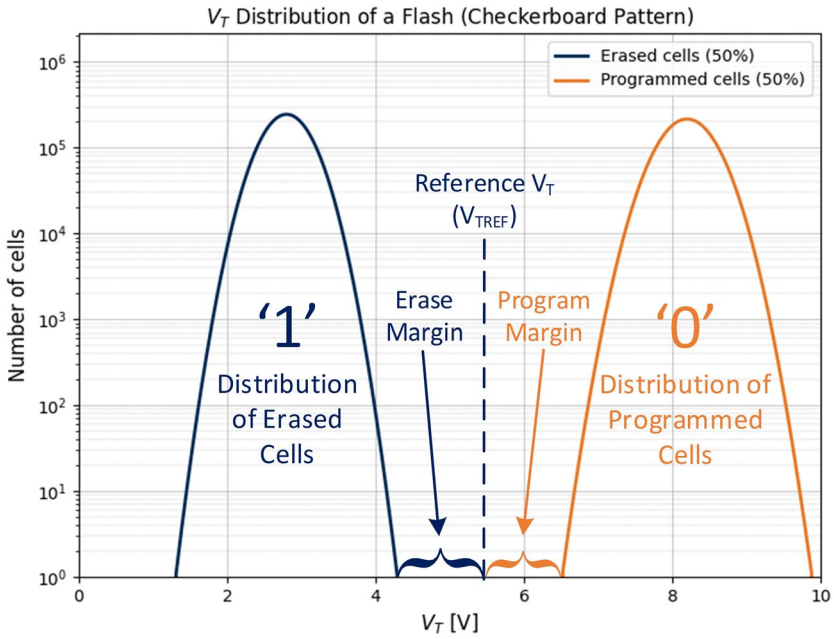

Figure 11 V_T Distribution Curve for a Checkerboard Pattern

For a device that is fully erased or programmed, the distribution curve contains only a single hump.

Note:

- the scale on the y-axis is logarithmic, and
- the humps have an approximate Gaussian Normal Distribution.

The middle of the V_T range between the erase and program is typically used as the reference threshold voltage (V_{TREF}) for Read operations, determining whether a particular group of cells are programmed or erased.

This voltage is applied to a selected Word-Line. In Figure 11. V_{TREF} is set to 5.5 V. This sets the Gate-Source voltage to V_{TREF} for the entire Page (see Figure 2). Depending on the state of the floating gate of each cell and which Bit-Lines are decoded/selected, an analog signal is output to the shared Bit-Lines. Sense Amplifiers and comparators are used to determine whether the output of each cell is a logical 1 or 0.

Cells that have a logical 1 as their output are the erased cells, and those whose output is logical 0 are the programmed ones.

Technically, this is how the normal Read operation is implemented in the chip.

The V_T Distribution Curve is obtained by sweeping the Word-Line voltage (WL) from 0 V to 10 V (typically), in 100 mV steps, while simultaneously applying 800 mV to the selected Bit-Lines. As the WL voltage sequentially increases, the number of cells that switch state from Erased to Programmed are recorded on the y-axis.

As the device cells are erased and programmed, the state of the cells (their V_T) is distributed. If we take a snapshot at any moment in time, we find that each device develops its own unique distribution. The distribution is dynamically changing during the lifetime of the device after each Program and Erase.

Toward the end of the lifetime of the device, the cells gradually lose their ability to be programmed and erased and ultimately get “stuck” in a narrow V_T range: they reach the end of their useful life.

For reliable operation, it is important to have a margin between programmed and erased cells, to ensure that over a prolonged usage the charges don’t flip from ‘0’ to ‘1’, or vice versa, as shown in Figure 11.

The integrated memory controller executes iterative algorithms that ensure for each program and erase cycle sufficient margins are reached. Between iterative erase/program operations, verification read operations are performed at specific targeted V_T voltages (“Pre-Program Verify,” “Erase Verify,” and “Recovery” lines on the V_T diagram).

5. Block Erase

During the lifetime of a NOR Flash, the device is erased and programmed many times. In the process the V_T distribution changes, because the state of individual cells is changing from Programmed to Erased and vice versa.

To maintain a nice V_T distribution with good margins, the Erase operation is performed in three distinct phases:

1. Pre-Program Phase
2. Erase Phase
3. Recovery Phase

As discussed earlier, Erase implies a state transition from ‘0’ to ‘1’. Performing a Block Erase over a large number of programmed cells on Figure 11 means shifting the Programmed cells (orange curve) to the left, to the position where the Erased cells (blue curve) currently reside.

Considering that the Erase operation affects all cells simultaneously, both humps (orange and blue curves) shift to the left.

The Erased cells (blue curve) would theoretically end up in the negative V_T territory. Physically, that is not possible, so, realistically, these cells become over-erased, with V_T close to 0 V, leaking current even when their Gate voltage is set to 0 V. This, of course, is bad.

To prevent this from happening, the already erased cells are first Pre-programmed. The objective is to create a distribution that is easier to manipulate as a single, compact group. Pre-program is done at the individual byte level; only cells that require it are pre-programmed, the rest are skipped over.

In the next sections, each of the three phases of the Erase operation is discussed in more detail.

5.1 Pre-Program Phase

The purpose of Pre-program is to create an ideal condition for the Erase operation. If a block of memory is erased without being pre-programmed, the cells already erased become over-erased.

To avoid creating over-erased cells, all erased cells must be Pre-programmed first. On the V_T diagram (see Figure 12) this is represented by shifting the distribution of the erased cells (dashed blue line) to the right, into the programmed region (dotted blue line).

After Pre-program, all cells inside the Flash device are programmed.

After Pre-program, the originally programmed cells (orange dotted line), and the pre-programmed cells (blue dotted line) produce a combined distribution represented by the solid orange curve in Figure 12.

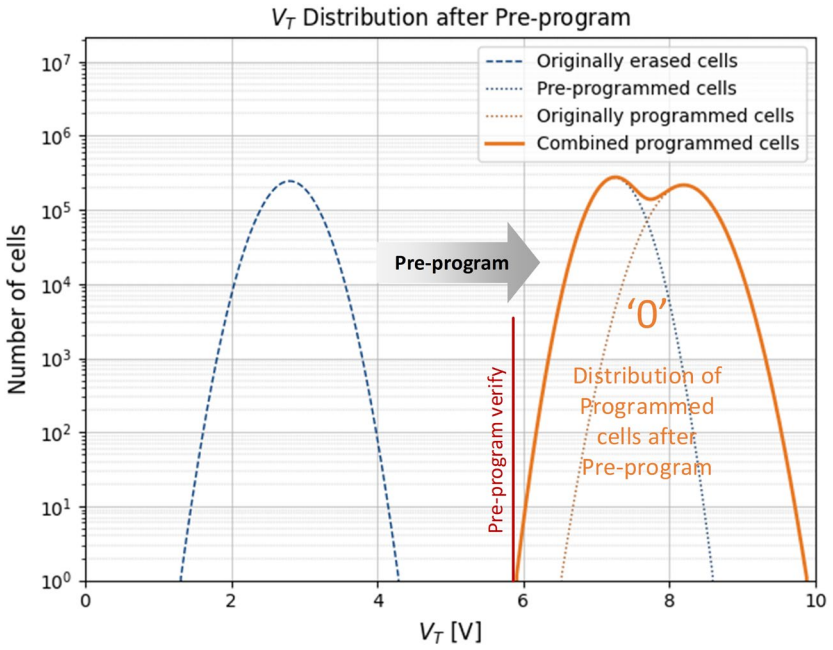

Figure 12 Pre-Programming Erased Cells

5.2 Erase Phase

The Erase of an individual cell was already explained in Section 3.3. Conceptually, the Erase Phase of a Block Erase is remarkably similar, except in this case instead of a single cell, an entire logical block is operated on simultaneously.

A large electrical field – created by a positive electrical voltage (+8 V) applied to the common p-well of the Physical Block, and a negative voltage (-10 V) applied to all Word-Lines within the Logical Block being erased – enable the electrons to “tunnel” through the potential barrier. The floating gates get increasingly depleted, lowering their V_T voltage as, shown in Figure 10.

This process is repeated with multiple Erase and Verify pulses, until the V_T of all cells moves below (to the left of) the Erase Verify Line. Graphically, this is represented by shifting the composite pre-programmed cell distribution to the left, as shown in Figure 13.

During this process, some cells can move close to the over-erased territory. As discussed above, such cells may conduct current, even when WL is inactive, and its V_T set to near 0 V.

Because it is done in bulk, an Erase must continue until the slowest cell verifies, the Erase operation can result in a large distribution.

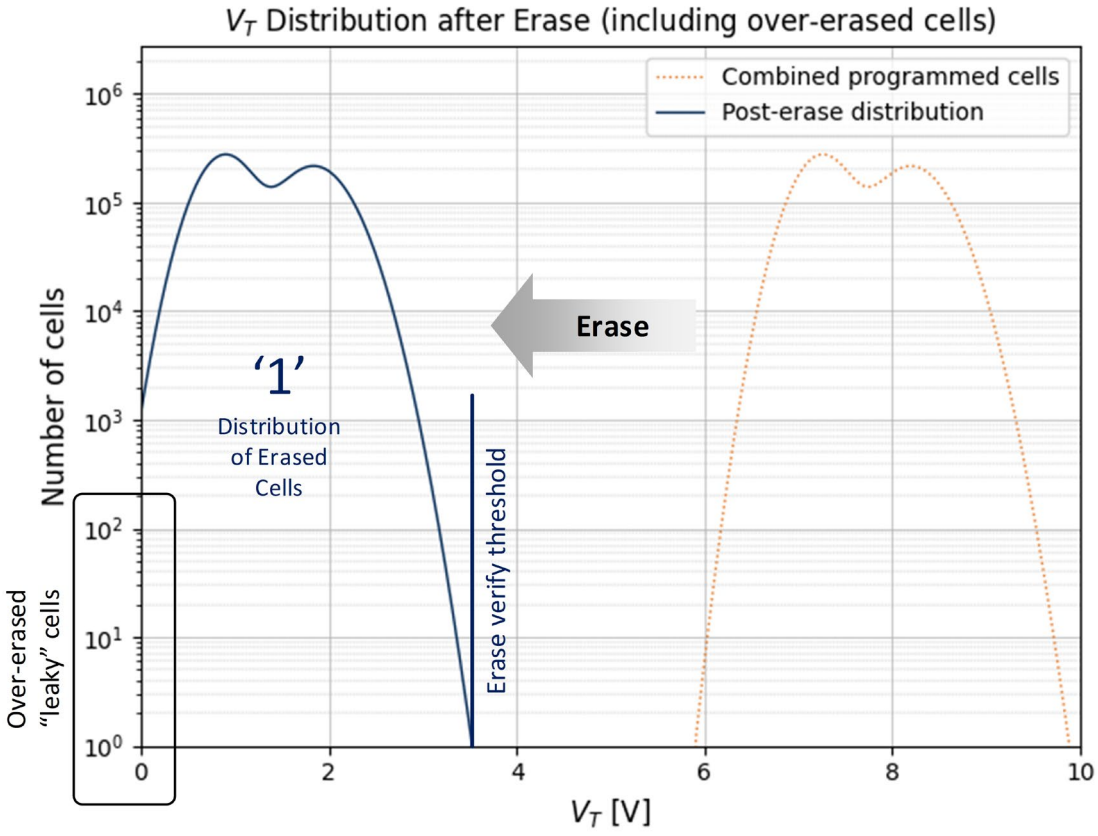

Figure 13 V_T Distribution After the Erase Phase

5.3 Recovery Phase

To deal with over-erased cells, they are systematically soft-programmed (on a byte-by-byte basis), until their VT shifts sufficiently to the right, so that it crosses the Recovery Line from the left, as shown in Figure 14.

Recovery adjusts the threshold of any bits over-erased during the Erase phase.

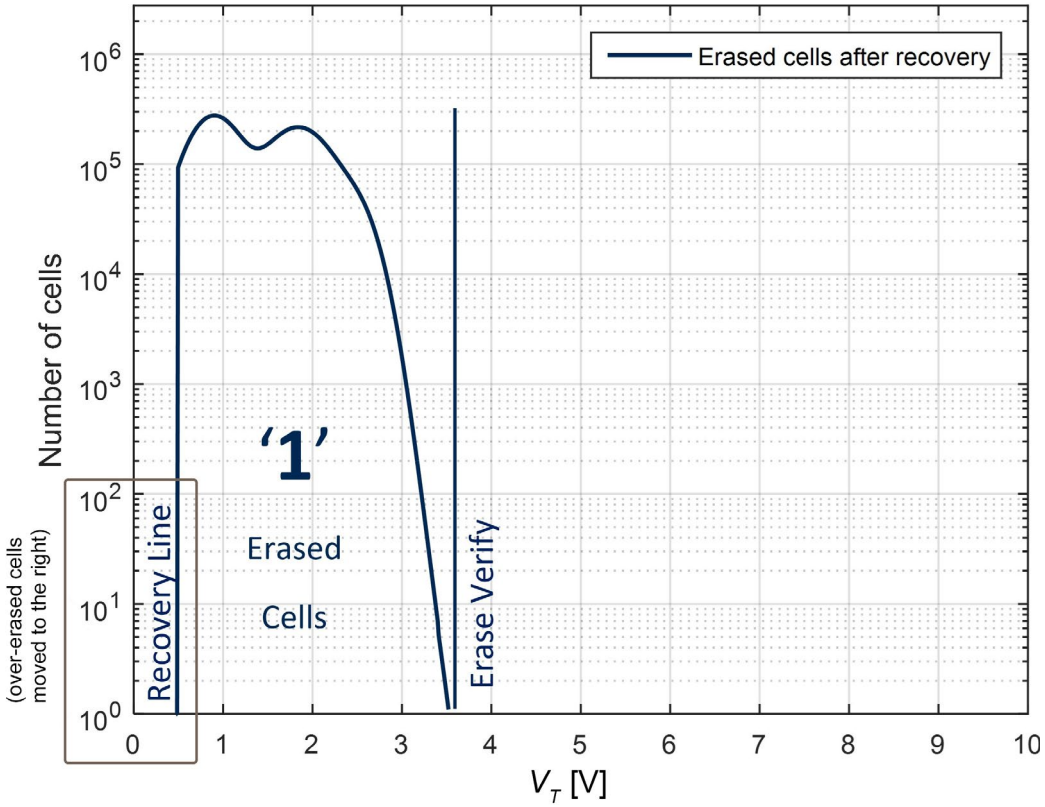

Figure 14 Over-Erase Compensation / Recovery Phase

While these internal algorithms are in progress, the Flash memory's integrated controller keeps a user-readable status-bit asserted (RDY#/BSY bit in the status register). This bit indicates whether an on-going internal operation has been completed or not.

The user **must not** reset the device, remove the power from the chip, and/or interrupt an already on-going Erase operation in any other manner while the RDY#/BSY bit is asserted.

6. Erase Interruption

If, for any reason, the Erase operation is interrupted intentionally or accidentally, the three phases of the erase operation cannot fully complete, and this may leave the memory in an undefined state.

6.1 Erase Interruption Types

An Erase operation can be interrupted for various reasons. At a high level, we can group these reasons as follows:

- Intentional interruption: this refers to the option provided by many flash products to suspend an erase operation. This is typically done when, during an erase operation, an interrupt occurs at the system level and the flash memory array must be read to serve that interrupt (for example: if the interrupt must execute code from the flash memory). Because an erase is a relatively long operation that blocks memory reads, and because a system interrupt must be typically served with low latency, the host is given the option to suspend the erase operation and resume it later.
- Asynchronous, possibly unintentional interruption: this refers to events like loss of power, power supply brown-out, or flash reset in the middle of an erase operation. In such cases, the three phases of the erase operation cannot fully complete, and this may leave the memory in an undefined state. The bits in the target block could be in any of the three states: Programmed, Erased, or Over-erased.

When events like this have occurred, it is safe to assume that the read data cannot be trusted.

6.2 Erase Interruption Scenarios

After an interruption of an erase operation, the state of the erased block is unknown to the host. The host should assume that the erase operation is not complete.

Obviously, the block to which the erase operation was applied is not ready for programming or reading. Those operations can be done only after a complete erase.

Also, depending on the state of the cells which are being erased, there is a very low probability that certain cells in the surrounding physical block may be read incorrectly.

Depending on which phase the Erase operation was in when the interruption occurred, multiple conditions are possible:

a. Interrupted the Pre-program phase -- some cells programmed, some erased.
b. Interrupted the Erase phase – some cells erased, some programmed, some over-erased.
c. Interrupted the Recovery phase – some cells erased, some over-erased.

For a., the corruption is obvious since the targeted block has nonsensical data.

For b. and c., the logical block being erased may appear fully erased, BUT, in fact, some of the bits may not have a sufficient erase margin or be over-erased. An insufficient margin can lead to early life failures of these bits.

Over-erased bits, as discussed earlier, result in leakage on the shared BL, which leads to reduced program margin for all other bits sharing the same physical BL. In this way, an interrupted erase can create a negative side effect for the entire shared Physical Block.

Figure 15 shows how a “leaky cell” on an unselected Word-Line can affect the reading of a “0 cell” on a selected Word-Line if there is a leaky cell on a shared Bit-Line.

In this example, there is an over-erased (leaky) cell on intersection of WL[65] and Bit-Line BL[1082]. The currently selected Word-Line is WL[241]. Although not likely, it may be possible for the amount of leakage (originating from the leaky cell on a shared Bit-Line) to be high enough so that the stored ‘0’ value in the selected cell (intersection of the Active Word-Line WL[65] and shared Bit-Line BL[1082]) is incorrectly read / detected as ‘1’.

If the number of leaky cells increases on a shared Bit-Line, due to the cumulative effect of the leakage currents, the probability of error (a ‘0’ on a selected Word-Line is erroneously read as ‘1’) increases.

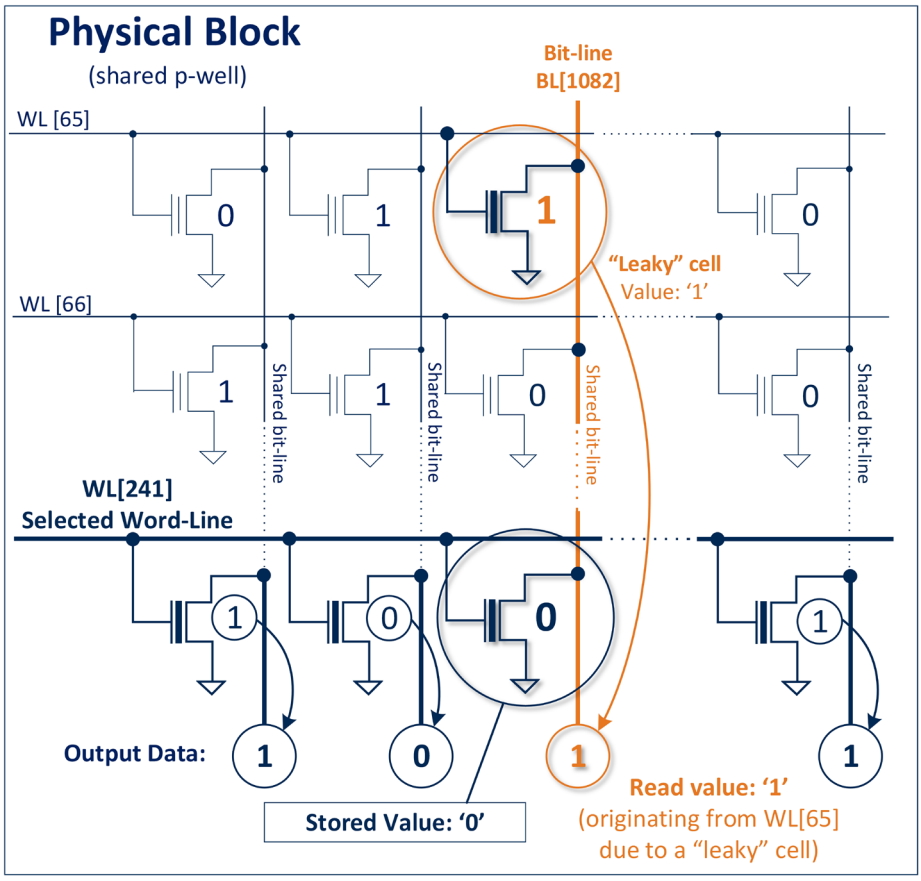

Figure 15 Data Corruption as Consequence of Over-Erased (“Leaky”) Cells

6.3 Remedies and Recommendations

To ensure a successful Erase operation and avoid the above potential failure scenarios, here are some recommendations:

1. Always check the RDY#/BSY status bit for any on-going operations before issuing a soft or hard RESET, and/or recycle the power supply. Failing to do so may lead to an undetected erase interruption, leaving the memory in an undefined state, resulting in data corruption.
2. Suspended erase operation: while an erase is suspended it is allowed to read from any memory address, but for reasons described above, it is recommended, despite the very low error probability, to avoid reading from the physical block in which the erase block resides.

   Example: assume that the physical block size is 2 Mbit or 256 kByte, or, in hex 0x40000 bytes. Also assume an erase operation has started at address 0x92000 and was suspended. While the erase is suspended it is best to avoid reading from the surrounding physical block at address range 0x80000-0xBFFFF. Anywhere else should be safe.

   In practice, it should not be difficult to implement this recommendation. An application typically divides the flash memory into partitions. If a system has n partitions, let’s mark them P0, P1, P2, …, Pn-1. The application developer should follow this rule: for example, if while writing to partition P2, the application must suspend an erase in order to read from partition P3, then P2 and P3 should not share the same physical block. If, on the other hand, there is no such conflict, then P2 and P3 can share a physical block. Once this

rule is followed and applied to any two bordering flash memory partitions, there is no risk of reading incorrect data during a suspended erase operation.

3. Asynchronous erase interruption: in this scenario the erase interruption is typically unintentional or unavoidable. In such a scenario, the flash loses track of the fact that it was in the middle of an erase operation because it has gone through a reset. The system must implement the following functions to address such a scenario:
   - Following a flash power-up or reset, detect the fact that an erase was interrupted. This detection function can be done in various ways which may or may not use the flash. The application should flag the start of an erase operation and not clear the flag until the erase is completed. It should also remember the address and size of the block which was being erased.
   - Recover from an incomplete erase: this requires the application to restart the erase operation which was interrupted. Once the restarted erase operation is complete, the recovery is done.

If you have further questions, Renesas Electronics Applications Engineering is available to provide answers and clarifications.

7. Revision History

Revision	Date	Description
A	Nov, 2019	Initial release.
A1	Feb, 2024	Applied new corporate template and layout. Numerous textual changes throughout. Read section significantly updated. Section 6 revised. Section 7 combined with Section 6.