Automatic ECC for Infineon MIRRORBIT™ architecture serial NOR flash memory families

About this document

Scope and purpose

AN200731 provides supplemental information regarding the Automatic ECC feature for Infineon MIRRORBIT™ architecture that is built into the Infineon FL/S-S, HL/S-T, FS-T, and KL/S-S serial NOR flash memory families.

Intended audience

This document is primarily intended for anyone who use the Infineon MIRRORBIT™ architecture serial NOR flash memory products who wants to understand and use the Automatic ECC feature for best device data intergrity.

Associated part families

FL/S-S, HL/S-T, KL/S-S, FS-T

Table of contents

1 Introduction

Reliability and data integrity are two important concerns of system designers on flash memory devices. Soft errors that are caused by high-energy particles striking semiconductor devices can result in bit flips, while continuous bit cost reduction (aggressive technology scaling) can deteriorate the storage reliability of flash memory. The current generation of automotive, industrial, or consumer systems have very low tolerance to data corruption and compromises reliability in flash memories. As a result, system designers have to build techniques such as off-chip error correction or software ECC, or implement redundancy to achieve higher reliability, which results in overhead.

Infineon NOR flash devices play a key role in several automotive applications such as advanced driver assistance systems (ADAS), instrument clusters, and infotainment. These critical applications require safe and reliable NOR flash devices to store code and data. Infineon' MIRRORBIT™ architecture-based serial NOR flash memory devices support embedded error detection and correction by generating a hamming error correction code (ECC) during memory array programming. This ECC is then used for error detection and correction during a read operation, which improves the overall functional safety of the systems. This application note provides supplemental information on the Automatic ECC feature that is built into the Infineon's FL/S-S, HL/S-T, FS-T, and KL/S-S flash memory families.

2 Automatic error correction code (ECC)

Automatic ECC performs internal hardware ECC generation, checking, and correction, which enhance the data integrity of flash devices. Error detection and correction (EDC) is applied to all parts of the flash address spaces other than registers. An ECC is calculated for each protected group of bytes and the ECC is stored in a hidden area related to the group of bytes. The following table summarizes ECC-related functions in serial NOR flash memory devices based on MIRRORBIT™ architecture.

Table 1 ECC functions

Function	Description
ECC / EDC Operation	ECC is calculated for each 16-byte aligned data.
	Single-bit EDC is supported with 8 ECC bits of hamming code per ECC unit, plus 1 bit for an ECC enable/disable flag.
	Configurable 2-bit detection feature is available. When 2-bit detection is enabled, any 1-bit error in an ECC unit is corrected and any 2-bit error is detected and reported.
	ECC is programmed as part of the standard program commands operation.
	To ensure the best data integrity is provided by EDC, each ECC unit should be programmed only once so that ECC is stored for that unit and not disabled. However, a second program operation on the same ECC unit disables ECC on that ECC unit; therefore, an erase operation is needed to re-enable the ECC enable/indicator bit for the ECC unit. It is recommended to program each 16-byte data once so that ECC remains enabled to provide the best data integrity.
	EDC protects the OTP region – however, a second program operation on the same ECC unit disables ECC permanently on that ECC unit (OTP is one-time programmable; therefore, an erase operation to re-enable the ECC enable/indicator bit is prohibited). It is recommended to program each 16-byte data once so that ECC remains enabled to provide the best data integrity.
	Calculation, programming, and disabling of ECC is done automatically as part of programming operations. Detection and correction if needed is done automatically as part of read operations. The host system sees only the corrected data from a read operation. Therefore, these are automatic operations transparent to the user.
ECC Enable/ Disable	ECC is disabled automatically if multiple programming operations are done on the same ECC unit.
	Single-byte programming or bit walking is allowed in certain flash device options, but it disables ECC on the second program to the same 16-byte ECC unit.
	To re-enable ECC for an ECC unit that has been disabled, the sector that includes the ECC unit must be erased. Sector erase resets all ECC Disable flags in a sector to the default state (enabled).
	A configuration register bit enables 1-bit error detection and correction, or 1-bit error detection and correction with 2-bit error detection.
ECC Status	The ECC Data Unit Status provides the status of ECC in each 16-byte data unit.
	The ECC Status Register provides the status indicating whether there has been 1-bit or 2-bit errors.
	The Address Trap Register provides the address of the first ECC error.
	The ECC Detection Counter keeps a tally of the number of 1-bit or 2-bit errors that have occurred in data units during reads.

Function	Description
	The INT# output flag in x8 devices can be enabled to indicate when either a 1- or 2-bit error detected as an ECC unit is read.
	The data strobe (DS) will be driven LOW (Stalled) upon 2-bit error detection

The following table summarizes the ECC-related functionality in Infineon's serial NOR flash devices with MIRRORBIT™ technology.

Table 2 Automatic ECC in Infineon's MIRRORBIT™ NOR flash devices

ECC functionality	FL-S, FS-S (x1, x2, x4)	S25HL-T, S25HS-T, S25FS-T (x1, x2, x4)	S28HL-T, S28HS-T (x8)	KL-S, KS-S (x8)	S26HL-T, 26HS-T (x8)
1-bit Detection and Correction	√	√	√	√	√
2-bit Detection	–	√	√	√	√
ECC Status Register	√	√	√	√	*
ECC Data Unit Status	*	√	√	*	√
Address Trap Register	–	√	√	√	√
ECC Detection Counter	–	√	√	√	√
INT# Output	–	–	√	√	√
Data Strobe (DS) Stall	–	–	–	√	√

* Functionality supported in another feature.

2.1 Programming with Automatic ECC

Figure 1 shows that ECC calculates syndrome bits for each 16-byte block and stores it in the hidden non-volatile flash memory. Based on the programming operation, it also sets the ECC Disable bit in the hidden area. As shown in **Figure 1**, there are 16 (or 32 blocks, depending upon the 256-byte or 512-byte page programming buffer option) blocks, each with its own unique ECC information, comprising a 256-byte (or 512-byte) line. The internal flash architecture also contains a 256-byte (or 512-byte) RAM structure called the Page Programming buffer. During a program operation, the Page Programming buffer is associated with a particular line of the flash memory array.

When data is first programmed within an ECC unit [1], the ECC value is set for the entire ECC unit. The 16-byte block is the smallest program granularity on which Automatic ECC is enabled. The Automatic ECC logic may affect data transfers from the Page Programming buffer to the line and is enabled or disabled based on the type and sequence of operations that have been applied on each block.

It is recommended that a multiple of 16-byte length and aligned blocks be written. This ensures that Automatic ECC is not disabled. Writing more than once to a 16-byte block after the prior erase can disable ECC on that block. It is recommended to program only once to any given block, then erase and repeat as necessary.

[1] The 16-byte aligned and length data groups in the main flash array and OTP array, each of which has its own hidden ECC syndrome to enable error correction on each group. This term refers to the union of the user-accessible bits in the flash array and the hidden ECC syndrome bits. Therefore, the group of protected bytes and the related ECC are together called an ECC unit.

Automatic error correction code (ECC)

There is a configuration option in a few flash memory devices to enable 2-bit error detection. When this option is enabled, any 1-bit error in an ECC unit is corrected and any 2-bit error is detected and reported. The 16-byte block requires a 9-bit ECC for 2-bit error detection. The ECC Disable bit used in multi-pass programming will be made configurable as the ninth ECC syndrome when 2-bit detection is enabled. In this case, multi-pass programming will be disabled; therefore, single-byte programming and bit walking in the same data unit that is programmed more than once is not allowed and will result in a program error.

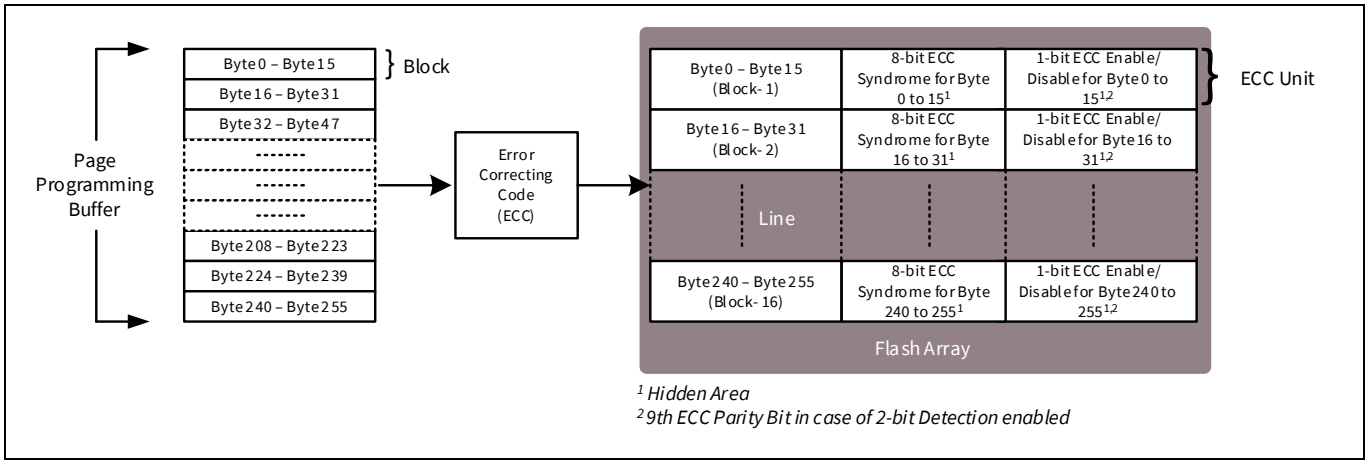


Figure 1 Logic block diagram for write path with Automatic ECC (256-byte page programming buffer example)

2.2 Read data with Automatic ECC

In combination with the EDC logic, ECC is used to detect and correct any single-bit error found during read access. During flash memory array read operations, a separate block-sized read buffer is used to temporarily hold information that is read from a block in the flash memory array and passed on to the user. The Automatic ECC logic may affect data transfers from the block to the read buffer, and is enabled or disabled based on the type and sequence of operations that have been applied to each block. When an ECC unit has Automatic ECC disabled, EDC is not done on data read from the ECC unit location.

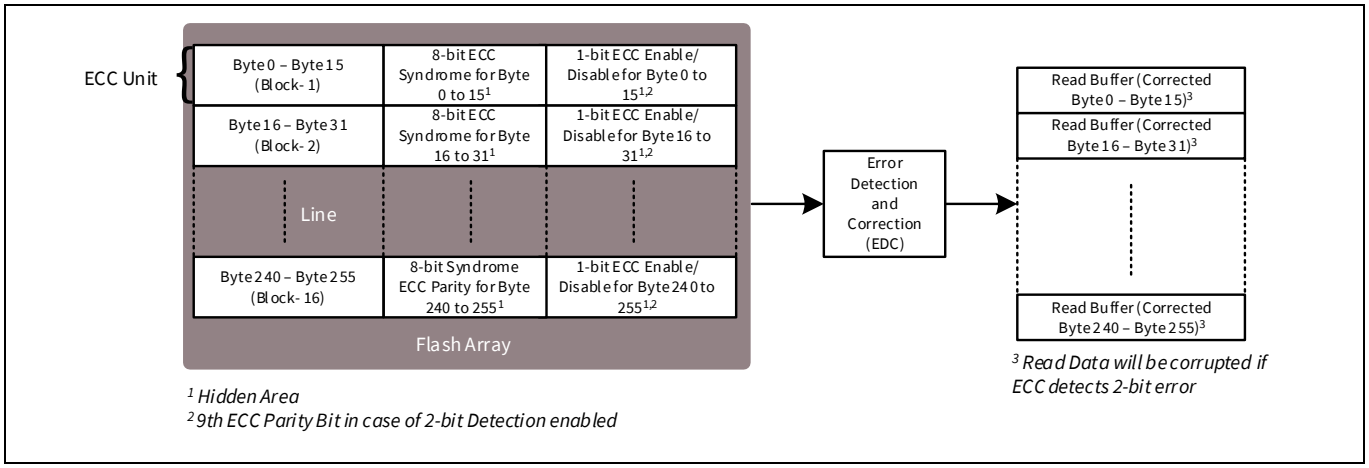


Figure 2 Logic block diagram for read path with Automatic ECC (256-byte page programming buffer example)

2.2.1 1-bit error detection and correction

This section demonstrates the flash device behavior when a 1-bit error occurs in the ECC unit.

Assume that you have a 16-byte data (all 0s) stored in the flash memory array with its 8-bit ECC syndrome (0x23 as an example) and ECC is always enabled. As shown in **Figure 3**, one bit from the 16-byte data is corrupted due to causes such as radiation or cell degradation. Now, the flash memory receives a read request for the particular address from the host. In this case, EDC detects and corrects the 1-bit error in data bytes before sending it out to the host.

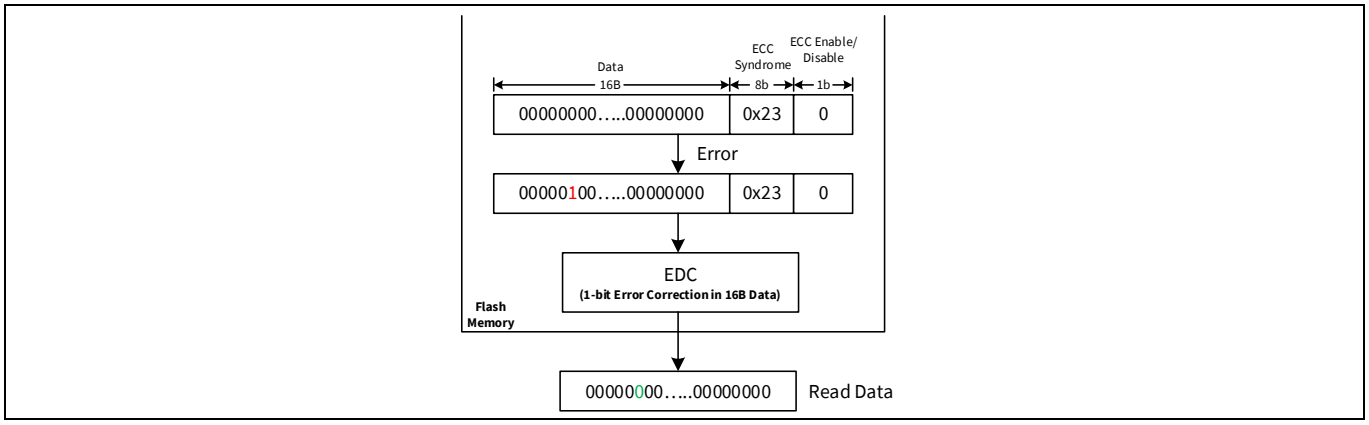


Figure 3 1-bit error correction example 1

Similarly, to send out the corrected read data to the host, the EDC can detect and correct a 1-bit error in the ECC syndrome as **Figure 4** shows.

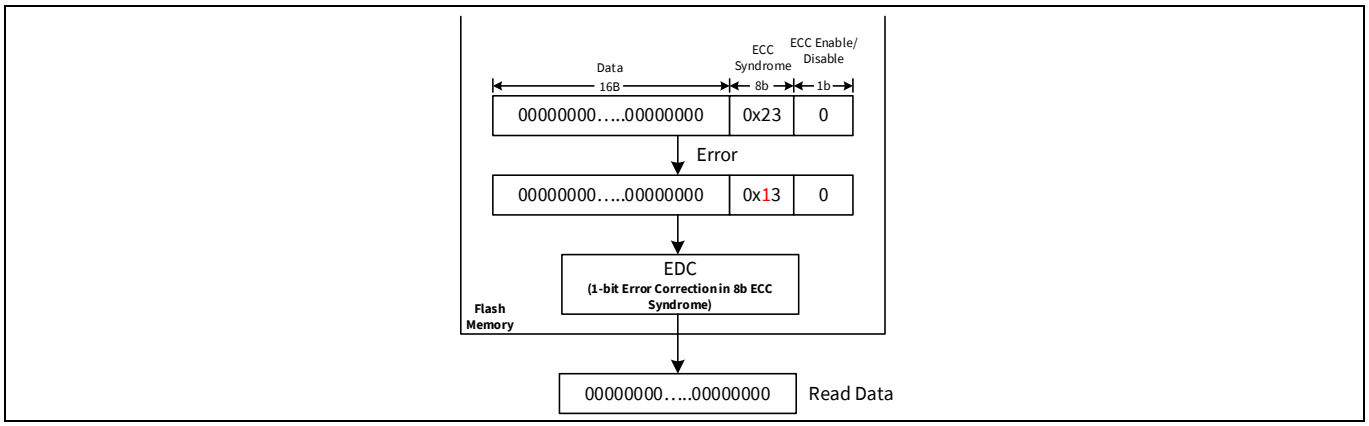


Figure 4 1-bit error correction example-2

2.2.2 2-bit error detection

This section demonstrates the flash device behavior when a 2-bit error occurs in the ECC unit.

Assume that you have a 16-byte data (all 0s) stored in the flash memory array with its 9-bit ECC syndrome (for example, all 1s). As shown in **Figure 5**, two bits from the 16-byte data is corrupted due to causes such as radiation or cell degradation. Now, the flash memory receives a read request for the particular address from the host. In this case, EDC detects and reports the 2-bit error in data bytes as **Figure 5** shows.

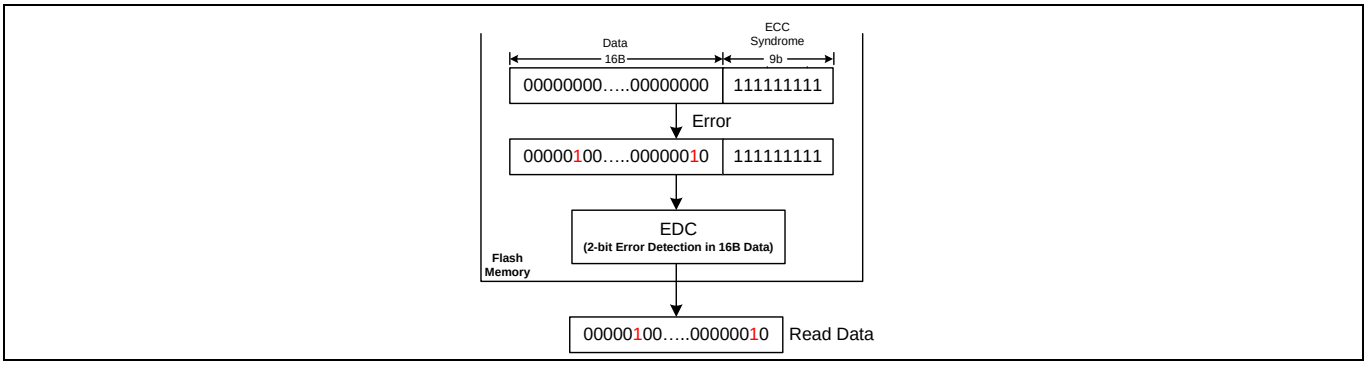


Figure 5 2-bit error detection example-1

If the flash device gets a 1-bit error in the 16-byte of data and a 1-bit error in the 9-bit ECC syndrome, EDC detects and reports the 2-bit error in the ECC unit as **Figure 6** shows.

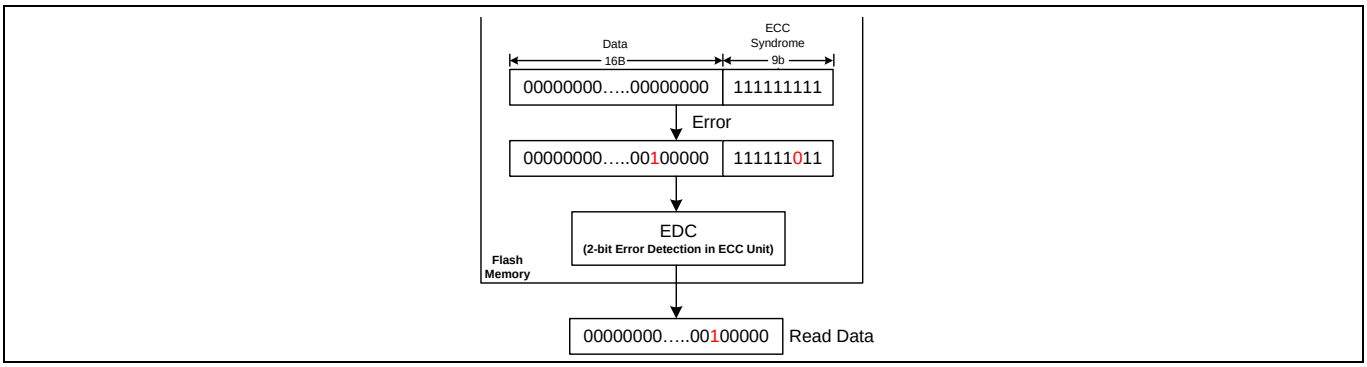
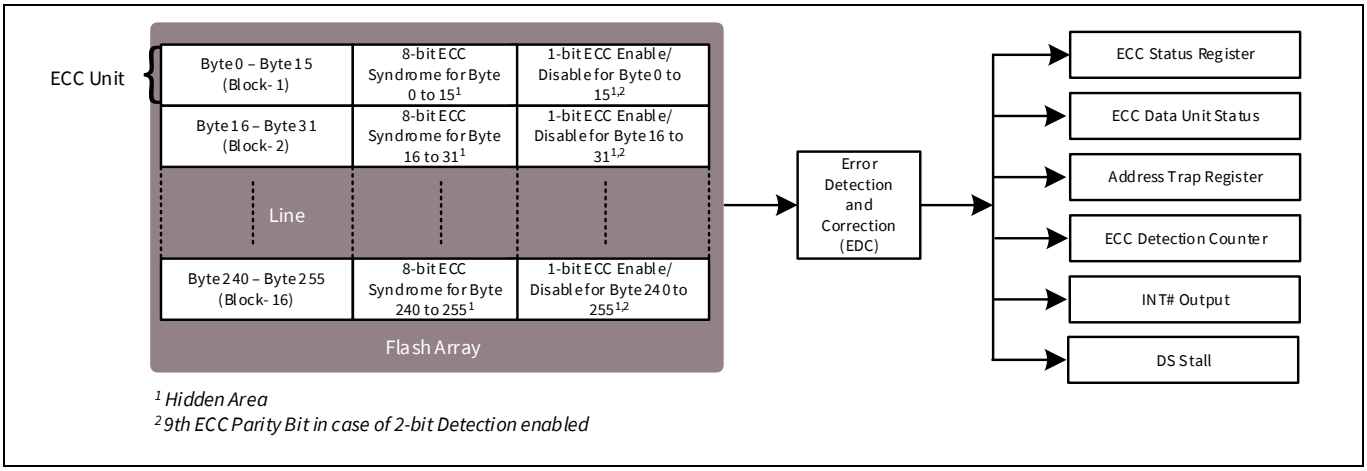


Figure 6 2-bit error detection example 2

2.3 ECC status

Figure 7 shows several methods for reporting to the host system when ECC errors are detected in Infineon's serial NOR flash devices. For more details on ECC Status commands and enablement, see the device datasheet.



[1] *Hidden Area*
[2] *9th ECC Parity Bit in case of 2-bit Detection enabled*

Figure 7 ECC status

2.3.1 ECC status register

Table 3 ECC status register functionality in Infineon NOR flash devices

Device	ECC status register functionality
FL-S, FS-S (x1, x2, x4)	Provides the status of 1-bit correction in 16-byte data contained in each ECC unit
	Provides the status of 1-bit correction in the 8-bit ECC syndrome contained in each ECC unit
	Provides the status of ECC enable/disable contained in each ECC unit
S25HL-T, S25HS-T, S25FS-T (x1, x2, x4)	Provides the status of 2-bit detection since last ECC status register clear. The respective register bit is updated every time any memory address is read and it remains set. The status is maintained until a clear ECC instruction is executed. The result regarding this bit is global and not dependent on any specific ECC unit address.
S28HL-T, S28HS-T (x8)	Provides status of 1-bit detection and correction since last ECC status register clear. The respective register bit is updated every time any memory address is read and it remains set. The status is maintained until a clear ECC instruction is executed. The result regarding this bit is global and not dependent on any specific ECC unit address.
KL-S, KS-S (x8)	Provides status of 2-bit detection since last ECC status ASO exit. The respective register bit is updated every time any memory address is read and it remains set. The status is maintained until the ECC status ASO exit instruction is executed. The result regarding this bit is global and not dependent on any specific ECC unit address.
	Provides status of 1-bit correction since last ECC status ASO exit. The respective register bit is updated every time any memory address is read and it remains set. The status is maintained until the ECC status ASO exit instruction is executed. The result regarding this bit is global and not dependent on any specific ECC unit address.
	Provides status of 1-bit correction in the 16-byte data contained in each ECC unit
	Provides status of 1-bit correction in the 8-bit ECC syndrome contained in each ECC unit
	Provides status of ECC enable/disable contained in each ECC unit
S26HL-T, S26HS-T (x8)	N/A (INT# supports this functionality)

2.3.2 ECC data unit status

Table 4 ECC data unit status functionality in Infineon NOR flash devices

Device	ECC data unit status functionality
FL-S, FS-S (x1, x2, x4)	N/A (ECC status register supports this functionality)
S25HL-T, S25HS-T, S25FS-T (x1, x2, x4) S28HL-T, S28HS-T (x8)	Provides status of 1-bit correction in each ECC unit
	Provides status of 2-bit detection in each ECC unit
	Provides status of ECC enable/disable in the selected ECC unit
KL-S, KS-S (x8)	N/A (ECC status register supports this functionality)
S26HL-T, S26HS-T (x8)	Provides status of 1-bit correction in each ECC unit
	Provides status of 2-bit detection in each ECC unit
	Provides status of ECC enable/disable in the selected ECC unit

2.3.3 Address trap register

Table 5 Address trap register functionality in Infineon NOR flash devices

Device	Address trap register
FL-S, FS-S (x1, x2, x4)	N/A
S25HL-T, S25HS-T, S25FS-T (x1, x2, x4) S28HL-T, S28HS-T(x8)	The address trap register stores the captured data unit address where either a 1-bit ECC correction or a 2-bit ECC detection occurred during a read operation. It stores the data unit address of the first ECC error captured during a single read operation.
KL-S, KS-S (x8)	The address trap register stores the captured data unit address where either a 1-bit ECC correction or a 2-bit ECC detection occurred during a read operation. It stores the data unit address of the first ECC error captured during a single read operation. The 512-Mbit HYPERFLASH™ density devices record only the address where a 2-bit error is encountered.
S26HL-T, S26HS-T(x8)	The address trap register stores the captured data unit address where either a 1-bit ECC correction or a 2-bit ECC detection occurred during a read operation. It stores the data unit address of the first ECC error captured during a single read operation.

2.3.4 ECC detection counter

Table 6 ECC detection counter functionality in Infineon NOR flash devices

Device	ECC detection counter
FL-S, FS-S (x1, x2, x4)	N/A
S25HL-T, S25HS-T, S25FS-T (x1, x2, x4) S28HL-T, S28HS-T (x8)	The ECC detection counter register stores the number of times ECC 1-bit corrections and ECC 2-bit detections have occurred during read operations since the last POR or hardware reset.
KL-S, KS-S (x8)	In HYPERFLASH™ devices other than the 512-Mbit density, a counter is provided to keep track of the number of 1-bit or 2-bit errors that occur as each block is read from the flash array. Only errors recognized in the main array (no active ASO) cause the error detection counter to increment. The counter does not increment while the ECC ASO is entered.
S26HL-T, S26HS-T (x8)	All HYPERFLASH™ lash devices support a counter to keep track of the number of 1-bit or 2-bit errors that occur as each block is read from the flash array. Only errors recognized in the main array (no active ASO) cause the error detection counter to increment. The counter does not increment while the ECC ASO is entered.

2.3.5 INT# output

Table 7 INT# output functionality in Infineon NOR flash devices

Device	INT# output
FL-S, FS-S (x1, x2, x4)	N/A
S25HL-T, S25HS-T, S25FS-T (x1, x2, x4)	N/A
S28HL-T, S28HS-T (x8)	These devices have an INT# pin, which is an open-drain output used to indicate to the host system that an event has occurred within the flash device. You can select to transition the INT# output pin to the active (LOW) state when any of the following occurs: 1. Transitioning from the Busy to the Ready state 2. 2-bit ECC error is detected 3. 1-bit ECC error is corrected
KL-S, KS-S (x8)	
S26HL-T, S26HS-T (x8)	

2.3.6 Data strobe stall

Table 8 Data strobe functionality in Infineon NOR flash devices

Device	Data strobe (DS) stall
FL-S, FS-S (x1, x2, x4)	N/A
S25HL-T, S25HS-T, S25FS-T (x1, x2, x4) S28HL-T, S28HS-T (x8)	N/A
KL-S, KS-S (x8)	These HYPERFLASH™ devices have a data strobe (DS) signal. If data strobe stall is enabled, upon double error detection, the DS will be driven LOW. DS will remain LOW as long as CS# remains asserted; normal DS functionality resumes as soon as CS# returns HIGH.
S26HL-T, S26HS-T (x8)	A mode may be enabled to cause the data strobe to stop toggling (stall) when reading a block containing a 2-bit error. The stall condition can be detected by the HYPERBUS™ master as a bus error when DS does not transition for more than 32 clock cycles. If enabled, upon 2-bit error detection, the DS is driven LOW after two clock cycles. DS will remain LOW as long as CS# remains asserted; normal DS functionality resumes as soon as CS# returns HIGH.

3 How flash operations affect ECC

Let us now look at how normal flash operations affect the ECC functionality with emphasis on the ECC information and the activity of the ECC and EDC logic blocks.

- The Sector Erase command changes each non-volatile flash bit in the sector to '1' or erased state. This operation affects all user-accessible bits within the sector, as well as the ECC information bits that are not visible to the user. The erased state of the ECC information does not enable EDC; therefore, when a Flash Memory Array Read request triggers the initial load of an erased block to the Read Buffer, EDC is not applied.
- Presuming that the following applies to a sector in the erased state, a subsequent program command copies the user-supplied bytes from the Page Program buffer to the appropriate line. Continuing the discussion for a single block taken from the set of blocks supplied by the user, the program command copies the user-supplied bytes from the Page Programming buffer to the corresponding block by changing '1' bits to '0' bits as necessary. The ECC parity information is computed for this block and is stored in the associated ECC information area together with an indication that EDC is enabled for this block; again, this is done by changing '1' bits to '0' bits, as necessary. When a subsequent Flash Memory Array Read request triggers the initial load of the block to the Read Buffer, EDC may be applied to correct at most 1-bit in either the block or its ECC information.
- Continuing from the non-erased state created by the program operation described in the previous scenario, if a subsequent programming operation of any type is applied to the block under consideration, the user data is copied from the Page Program buffer to the block by changing '1' bits to '0' bits as necessary. Note that '0' bits from any prior programming operation are not changed to '1' bits by this new operation because a program command cannot perform this transition. Incremental ECC parity is computed; if the new ECC parity requires any forbidden '0' bit to '1' bit transitions, the ECC information is updated to disable the EDC logic for that entire block. On the other hand, if only '1' bit to '0' bit transitions are required, then incremental parity bits are programmed; ECC remains enabled for that block. When EDC is disabled for a block, it remains disabled for subsequent programming operations of any type and cannot be enabled until the entire sector is erased. When a subsequent Flash Memory Array Read request triggers the initial load of this block to the Read Buffer, EDC is not applied.
- Interrupted program/erase operations can lead to user data bits that are inconsistent with the hidden parity bits; these appear as uncorrectable bit errors when the ECC registers are queried. So, if the system reads ECC registers, it must ensure that the ECC Data Units scanned have not been subjected to interrupted program or erase. The Evaluate Erase Status command can be used to check sectors for interrupted erase. System-provided methods must be used to identify locations that may have been subjected to interrupted erase. ECC Register reads on ECC Data Units that have been programmed/erased to completion are trustworthy.
- Single-bit errors within a single ECC Data Unit are correctable, so they are never seen by the system. When single-bit errors are observed while scanning the array using the ECC register reads, these errors are not faults, and no further action is required by the system. Two-bit errors within a single ECC Data Unit are not correctable, so they may affect system operation. Reading the ECC registers can identify the location of two-bit errors, thereby enabling the system to mitigate the error. Devices with HYPERBUS™ or Octal SP interfaces may provide an interrupt signal if ECC errors are detected during normal read operations; this enables the system to check for 2-bit errors and take corrective action before the bit error affects system operation.

Based on this discussion, we can infer that the Infineon flash devices with automatic ECC are backward-compatible with all your existing flash applications. This means that the new Automatic ECC feature is transparent to the ways in which you use flash memory today. In addition, if your application demands it, you can enable Automatic ECC for an erased sector by always writing only once to any given block, and then erasing and repeating as necessary.

The following sections address the two methods that Infineon recommends to enable your application for high-reliability usage. Normal applications may use software logic that disables the Automatic ECC feature for some regions of the array – you may continue to use these applications without change.

3.1 High-reliability usage: 100% ECC Fraction

In the following cases, the best practice is to program full lines for the entire span of the programmed region, taking care to ensure that no line is programmed more than once after the prior erase:

- Factory-programming the initial factory image
- Field-reprogramming any element of the Binary and Data Partitions
- Operating applications over any element of the Binary and Data Partitions

This practice delivers the fastest programming performance, and ensures that the Automatic ECC feature is enabled for every block within the programmed region.

If your factory programmer or field update programmer needs to program some flash regions at granularities smaller than a 512-byte line, the same effect with respect to Automatic ECC can be obtained by programming granularities no smaller than a 16-byte block, again taking care to ensure that no block is programmed more than once after the prior erase. Similarly, for the Data Partition, the best practice is for your file system software to program at granularities no smaller than a 16-byte block, where the software ensures that no block is programmed more than once after the prior erase. These programming practices ensure that the flash array is used with 100% ECC Fraction, where ECC Fraction is defined as follows:

$$\text{ECC Fraction} = \frac{\text{No. of Blocks with ECC Enabled}}{\text{No. of Blocks in the flash device}}$$

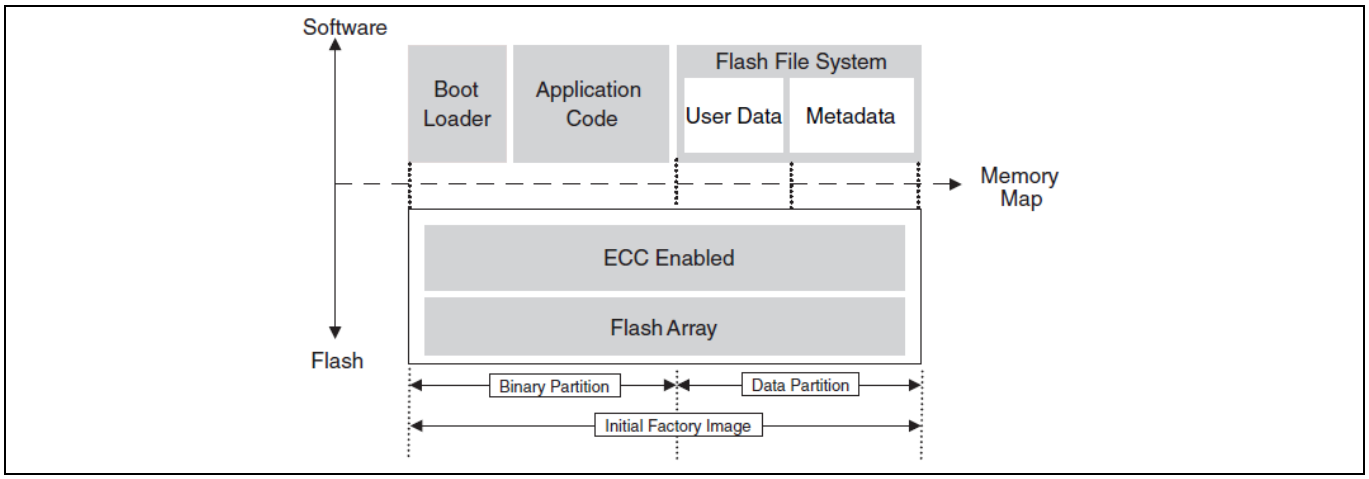


Figure 8 Reference memory map for the case of 100% ECC Fraction

The main challenge with achieving 100% ECC Fraction is that not many flash file systems or block drivers for NOR or SPI flash are designed to operate in single-pass programming mode. NAND file systems comply with this operating mode because the NAND device specifications require it. It is possible to convert a NAND file system or block driver to run on NOR or SPI flash so that the flash can be operated with 100% ECC Fraction.

3.2 High-reliability usage: 100% effective ECC Fraction

As **Figure 9** shows, the Binary Partition is managed with 100% ECC Fraction according to the methods described in the previous section. The Data Partition is managed by a normal NOR or SPI file system or block

driver that has been upgraded with added redundancy to achieve 100% effective ECC Fraction for this region. Effective ECC Fraction is defined as follows:

$$\text{Effective ECC Fraction} = \frac{\text{No. of Blocks with ECC Enabled + Mitigated Blocks}}{\text{No. of Blocks in the flash device}}$$

In this definition, a 'Mitigated Block' is a block where flash operations have disabled or will disable the Automatic ECC function for that block, but the software provides additional redundancy that replaces the benefit of the disabled Automatic ECC function.

The basic concept underlying this approach is that the overall system does not care whether the data storage subsystem manages data integrity in hardware or software – this approach simply replaces the disabled Automatic ECC with alternative redundancy that is managed by the software for all sub-block writes. Any redundancy method can be used so long as you ensure that the method can return the correct data if a single bit error happens within the metadata element stored within a block. This approach can be less disruptive to your trusted software stack because only a minor patch needs to be re-qualified: the core algorithms of your current NOR file system are unaffected. However, backward compatibility with your existing on-flash formats is affected because the on-flash metadata structures are changed as a result of this approach.

The discussion that follows uses two examples to illustrate this approach. These are not the only possible methods, but they provide you with a good starting point for adapting your file system or block driver to 100% effective ECC Fraction.

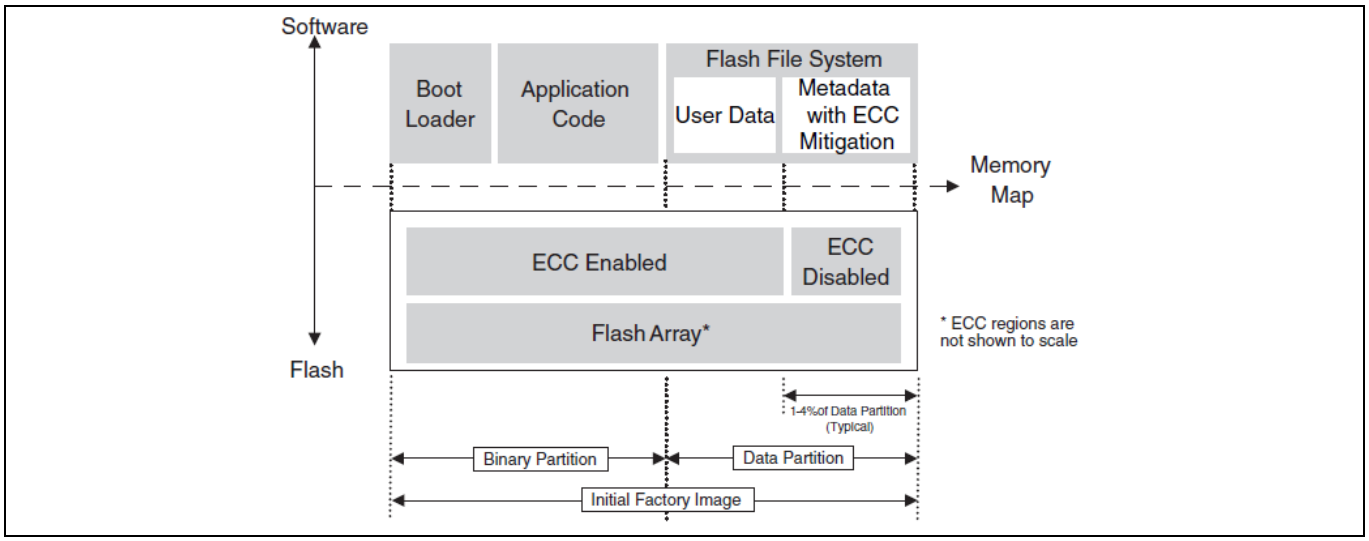


Figure 9 Reference memory map for the case of 100% effective ECC Fraction

3.2.1 Example 1: Software ECC

This example is relevant for logging algorithms wherein you have a simple list of metadata elements, all smaller than a block, where each metadata element must be written in a single pass but at different times during the execution of your data storage algorithm. This method simply appends a few bits of the ECC data to each metadata element, where these extra bits are written to the flash at the same time as the metadata element. It is clear that your algorithm cannot overwrite these regions, because this action may invalidate the software ECC just as it might for the hardware method. **Figure 10** shows the method in more detail for the case of 'N' metadata elements stored within a single block.

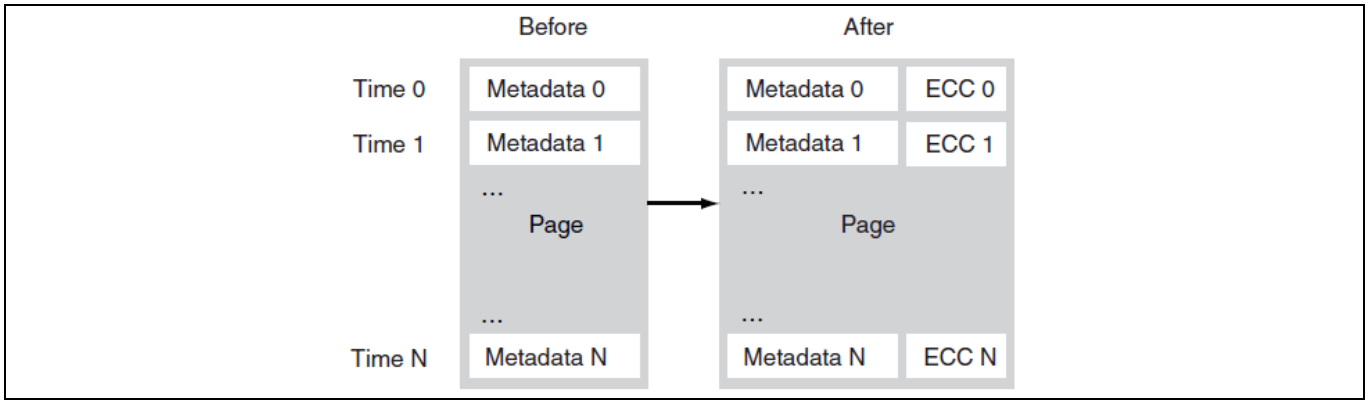


Figure 10 Mitigating disabled Automatic ECC with Software ECC

As the most direct approach of using software to replace the disabled Automatic ECC, this approach delivers correct data in the event of a single-bit error in either the metadata element or its associated ECC data. Data redundancy overhead is minimal with this approach in that the software ECC requires the fewest number of extra bits to achieve the requisite level of bit error protection. Software encode/decode overhead is theoretically larger for this method compared to others, but in practice the observable programming overhead is quite small, mainly because the size of the data under software ECC protection is small. Infineon prototypes using this method confirm that the effect on both data storage efficiency and programming overhead can be virtually undetectable. In contrast to programming, the additional software overhead incurred by decoding the software ECC on the metadata may noticeably degrade the effective read speed.

3.2.2 Example 2: Three-Copy Voting

In contrast to the Software ECC method, the Three Copy Voting method is appropriate for cases where your algorithm overwrites a metadata storage location one or more times to track a progression of states via bit walking. This method takes a single copy of the data element that is overwritten multiple times, and transforms it to three identical copies, which are then overwritten in triplicate for each update to the metadata element. Upon reading the metadata element later, if any two of the metadata copies agree, then the value contained in these two identical copies is passed to the application as the 'true' value of the metadata element.

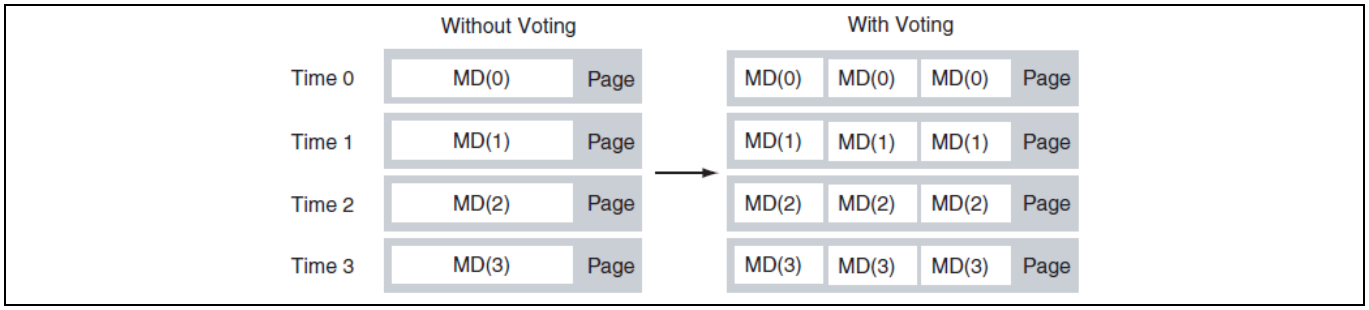


Figure 11 Mitigating disabled Automatic ECC with Three-Copy Voting

As **Figure 11** shows, each bit of the data content in the original metadata element, "MD", is replaced with a new metadata element comprising three copies of "MD" that contains three bits of storage for each bit of data content in the original single copy of the metadata element. If any one of these three bits is erroneous, the method returns the 'true' value of that bit. When this method is applied bit-wise to the three copies of the metadata element, even if one copy of every bit in the three-copy metadata element is erroneous, this method still returns the 'true' value of the metadata element. Because the bit-wise approach delivers much more

redundancy than is actually required to replace the disabled Automatic ECC, software overhead can be minimized by decoding the metadata value via the copy-wise method as originally suggested above.

When the copy-wise decoding method is used, the Three-Copy Voting method can withstand a single-bit error within the scope of the three copies of that metadata element. Data redundancy overhead per metadata element is rather large at 300%, but these types of metadata elements are normally quite small (word-sized), so in the grand scheme of the Data Partition, the net impact is likely to be small. Software overhead for a copy-wise decoding algorithm is also very small, so the read speed impact is small, but potentially noticeable.

4 Summary

The 65-nm and 45-nm serial NOR flash devices from Infineon have an Automatic ECC feature that is completely transparent for normal modes of flash operation. Nearly all existing consumer and industrial applications can migrate to the flash without any special software considerations. For your automotive or other high-reliability applications, this application note also shows how you can adapt your application to achieve either 100% ECC Fraction, or 100% Effective ECC Fraction.

Revision history

Document version	Date of release	Description of changes
**	2012-04-06	Initial release
*A	2012-07-18	Updated section: Automatic ECC Programming Guide: Logic Diagram for Read and Write Paths with Automatic ECC figure: changed Block 15 to Block 32
*B	2015-12-09	Updated in template
*C	2016-06-29	Updated the AN to support FS-S in addition to FL-S. Updated template
*D	2017-08-02	Updated logo and copyright.
*E	2018-04-26	Updated product information
*F	2022-06-02	Updated to Infineon template Added S25FS-T device

Trademarks
All referenced product or service names and trademarks are the property of their respective owners.

Edition 2022-06-02

Published by

Infineon Technologies AG

81726 Munich, Germany

Do you have a question about this document?

Go to www.infineon.com/support

Document reference

002-00731 Rev. *F

IMPORTANT NOTICE

The information contained in this application note is given as a hint for the implementation of the product only and shall in no event be regarded as a description or warranty of a certain functionality, condition or quality of the product. Before implementation of the product, the recipient of this application note must verify any function and other technical information given herein in the real application. Infineon Technologies hereby disclaims any and all warranties and liabilities of any kind (including without limitation warranties of non-infringement of intellectual property rights of any third party) with respect to any and all information given in this application note.

The data contained in this document is exclusively intended for technically trained staff. It is the responsibility of customer's technical departments to evaluate the suitability of the product for the intended application and the completeness of the product information given in this document with respect to such application.

For further information on the product, technology, delivery terms and conditions and prices please contact your nearest Infineon Technologies office (**www.infineon.com**).

WARNINGS

Due to technical requirements products may contain dangerous substances. For information on the types in question please contact your nearest Infineon Technologies office.

Except as otherwise explicitly approved by Infineon Technologies in a written document signed by authorized representatives of Infineon Technologies, Infineon Technologies' products may not be used in any applications where a failure of the product or any consequences of the use thereof can reasonably be expected to result in personal injury.