AN200731

Automatic ECC for Cypress 65-nm MirrorBit® Eclipse™ SPI Flash Nonvolatile Memory Families

This application note provides supplemental information regarding the Automatic ECC feature that is built into the FL-S and FS-S MirrorBit Eclipse flash families.

1 General Description

This application note provides supplemental information regarding the Automatic ECC feature that is built into the FL-S and FS-S MirrorBit Eclipse flash families. The information contained in this document modifies any information on the same topics established by the data sheets listed in section 5, *Related Documents* on page 11 and should be used in conjunction with those documents. This document may also contain information that was not previously covered by the FL-S and FS-S data sheets. It is intended for hardware system designers and software developers of applications, operating systems, or tools.

2 Device Description

2.1 Distinctive Characteristics

- Programming
  - Automatic ECC — internal hardware Error Correction Code generation, checking, and correction.

2.2 New Features

- Automatic ECC for enhanced data integrity.

2.3 Glossary

- ECC Unit = 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.
- Program Block = this is the user-accessible 16 byte aligned portion of an ECC Unit.

2.4 Error Correction Code (ECC)

Error Detection and Correction (EDC) is applied to all parts of the flash address spaces other than registers. An Error Correction Code (ECC) is calculated for each group of bytes protected and the ECC is stored in a hidden area related to the group of bytes. The group of protected bytes and the related ECC are together called an ECC unit.

- ECC is calculated for each 16-byte aligned and length ECC unit.
- Single Bit EDC is supported with 8 ECC bits of hamming code per ECC unit, plus 1 bit for an ECC disable Flag.
- Sector erase resets all ECC disable flags in a sector to the default state (enabled).
- ECC is programmed as part of the standard Program commands operation.
- ECC is disabled automatically if multiple programming operations are done on the same ECC unit.
- Single-byte programming or bit walking is allowed but disables ECC on the second program to the same 16-byte ECC unit.

- The ECC disable flag is programmed when ECC is disabled.
- To re-enable ECC for an ECC unit that has been disabled, the Sector that includes the ECC unit must be erased.
- To ensure the best data integrity provided by EDC, each ECC unit should be programmed only once so that ECC is stored for that unit and not disabled.
- The calculation, programming, and disabling of ECC is done automatically as part of programming operations. The detection and correction if needed is done automatically as part of read operations. The host system sees only corrected data from a read operation.
- There is a command to read the ECC status for any ECC unit.
- 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, hence an erase operation to re-enable the ECC enable/indicator bit is prohibited).

2.5 ECC Status Register (ECCSR)

Related Commands for FL-S family: ECC Read (ECCRD 18h).

Related Commands for FS-S family: ECC Read (4ECCRD 18h and ECCRD 19h).

The status of ECC in each ECC unit is provided by the 8-bit ECC Status Register (ECCSR). The ECC Register Read command is written followed by an ECC unit address. The contents of the status register then indicates, for the selected ECC unit, if ECC is disabled for that ECC unit. [1]

Table 1. ECC Status Register (ECCSR)

Bits	Field Name	Function	Type	Default State	Description
7 to 3	RFU	Reserved		0	Reserved for Future Use
2	EECC	Single Bit Error corrected in ECC Syndrome Data	Volatile, Read only	0	1 = single bit error corrected 0 = no error corrected
1	EECCD	Single Bit Error corrected in Program Block	Volatile, Read only	0	1 = single bit error corrected 0 = no error corrected
0	ECCDI	ECC is disabled for the ECC Unit	Volatile, Read only	0	1 = ECC is disabled 0 = ECC is enabled

ECCSR[0] = 1 indicates that ECC is disabled for the ECC Unit, while ECCSR[0]=0 indicates that ECC is enabled for the ECC Unit. Note that ECC is only disabled for an ECC Unit if the next ECC value cannot be achieved by changing ones in the prior ECC value to zeros; thus ECC may remain enabled even after multiple writes, including word writes, to a Program Unit. This means ECCSR[0] is a reliable indicator for whether or not ECC logic will be applied to the read path for that Program Unit.

The ECC for each ECC unit covers both the data programmed by the user and the syndrome calculated by the flash. If the read of an ECC unit requires correction of a single bit in data programmed by the user, then ECCSR[1]=1 and ECCSR[2]=0. If the read of an ECC unit requires correction of a single bit in the ECC syndrome data, then ECCSR[1]=0 and ECCSR[2]=1. If the read of an ECC requires no correction, then ECCSR[1]=0 and ECCSR[2]=0.

The bits ECCSR[7:3] are reserved. These bits have undefined high or low values that can change from one ECC status read to another. These bits should be ignored by any software reading The ECC Status Register.

2.6 Programming OTP Memory Space

Automatic ECC is programmed on the first programming operation to each 16-byte region. Programming within a 16-byte region more than once disables the ECC. It is recommended to program each 16-byte portion of each 32-byte region once so that ECC remains enabled to provide the best data integrity.

1. The Read Any Register (RDAR 65h) command supported by the FS-S family cannot be used to read the ECC Status Register (ECCSR).

2.7 Command Summary by Function

2.7.1 FL-S Command Summary by Function

Table 2. FL-S Family Command Set — Sorted by Function

Function	Command Name	Command Description	Instruction Value (Hex)	Maximum Frequency (MHz)
Register Access	ECCRD	ECC Read	18	133

2.7.2 FS-S Command Summary by Function

To accommodate addressing above 128 Mb, the FS-S family has two command options for ECC Read:

1. The 4ECCRD (18h) command always requires a 4-byte address and is used to access up to 32 Gb of memory.
2. The ECCRD (19h) command is used for backward compatibility to devices with 3-byte address instructions. ECCRD can be used in conjunction with a 4-byte address mode controlled by the Volatile Configuration Register 2 Bit 7 (CR2V[7]). See the datasheet for more information on extended addressing. The ECCRD command has a variable latency which can be selected in the Nonvolatile Configuration Register 2 Bits 3 to 0 (CR2NV[3:0]) and adjusted during normal operation in the Volatile Configuration Register 2 Bits 3 to 0 (CR2V[3:0]). See the datasheet for more information on selecting the variable latency.

Table 3. FS-S Family Command Set - Sorted by Function

Function	Command Name	Command Description	Instruction Value (Hex)	Maximum Frequency (MHz)	Address Length (Bytes)
Register Access	4ECCRD	ECC Read	18	133	4
Register Access	ECCRD	ECC Read	19	133	3 or 4

2.7.2.1 ECC Read During Erase or Program Suspend

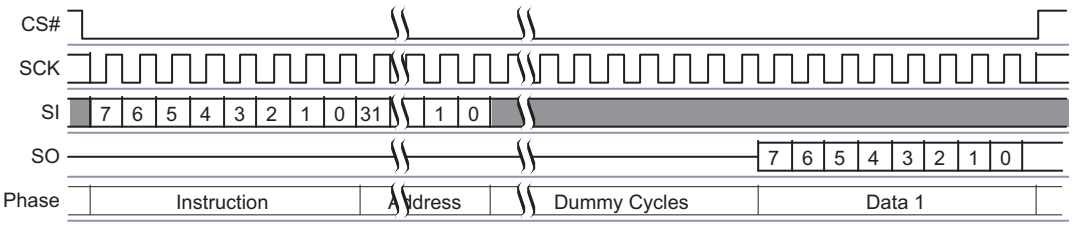
The 4ECCRD and ECCRD commands are allowed during erase or program suspend operations.

2.8 ECC Status Register Read

2.8.1 FL-S ECC Status Register Read (ECCRD 18h)

To read the ECC Status Register, the command is followed by the 32-bit ECC unit address, the four least significant bits (LSB) of address must be set to 0. This is followed by eight dummy cycles. Then the 8-bit contents of the ECC Register, for the ECC unit selected, are shifted out on SO 16 times, once for each byte in the ECC Unit. If CS# remains low the next ECC unit status is sent thru SO 16 times, once for each byte in the ECC Unit. The maximum operating clock frequency for the ECC READ command is 133 MHz.

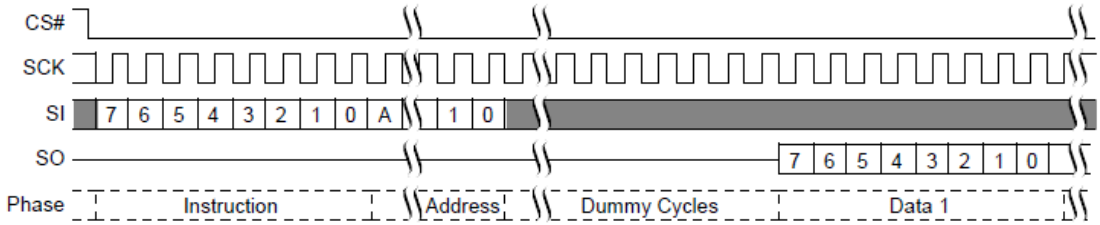
Figure 1. ECC Status Register Read Command Sequence



2.8.2 FS-S ECC Status Register Read (ECCRD 19h or 4ECCRD 18h)

To read the ECC Status Register, the command is followed by the ECC unit address, the four least significant bits (LSB) of address must be set to zero. This is followed by the number of dummy cycles selected by the read latency value in CR2V[3:0]. Then the 8-bit contents of the ECC Register, for the ECC unit selected, are shifted out on SO 16 times, once for each byte in the ECC Unit. If CS# remains low, the next ECC unit status is sent thru SO 16 times, once for each byte in the ECC Unit. The maximum operating clock frequency for the ECC READ command is 133 MHz.

Figure 2. ECC Status Register Read Command Sequence



Note

1.A = MSB of address = 23 for Address length CR2V[7] = 0, or 31 for CR2V[7]=1 with command 19h.

2.A = MSB of address = 31 with command 18h.

ECCRD and 4ECCRD are supported in Quad All mode. In Quad All mode the instruction is shifted in on IO0-IO3, two clock cycles per byte. The ECC read command is not supported for 0 dummy latency in Quad All mode.

Figure 3. ECCRD (19h), Quad All Mode, Configuration Register 2 Bit 7 (CR2[7]) = 0, Command Sequence

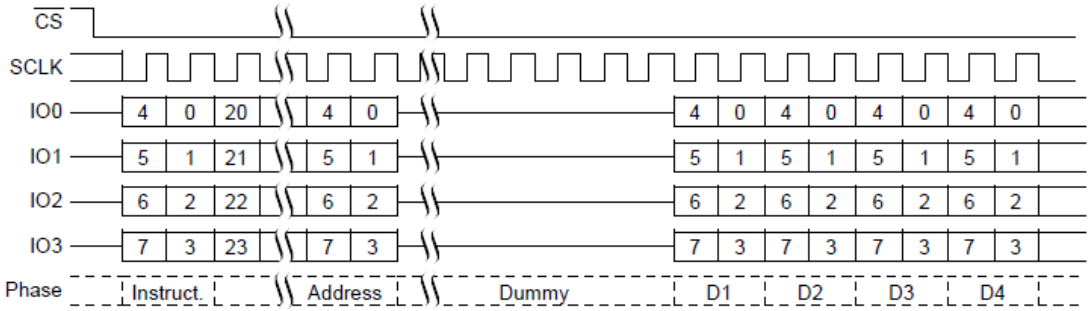
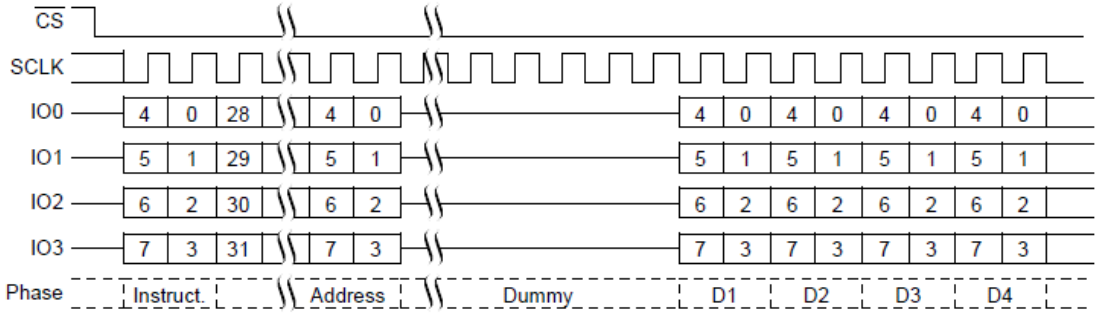


Figure 4. ECCRD (19h), Quad All Mode, CR2[7] = 1, or 4ECCRD (18h) Command Sequence



2.9 Automatic ECC

Each 16-byte aligned and 16-byte length Program Block has a hidden Error Correction Code (ECC) value. The Program Block plus ECC form an ECC unit. In combination with Error Detection and Correction (EDC) logic the ECC is used to detect and correct any single bit error found during a read access. When data is first programmed within an ECC unit, the ECC value is set for the entire ECC unit. If the same ECC unit is programmed more than once the ECC value is changed to disable the EDC function. A sector erase is needed to again enable Automatic ECC on that Program Block. The 16-byte Program Block is the smallest program granularity on which Automatic ECC is enabled.

These are automatic operations transparent to the user. The transparency of the Automatic ECC feature enhances data accuracy for typical programming operations that write data once to each ECC unit, but facilitates software compatibility to previous generations of FL and FS families of products by still allowing for single-byte

programming and bit walking in which the same ECC unit is programmed more than once. When an ECC unit has Automatic ECC disabled, EDC is not done on data read from the ECC unit location.

An ECC status register is provided for determining if ECC is enabled on an ECC unit. The ECC Status Register Read (ECCRD and 4ECCRD) commands are used to read the ECC status on any ECC unit.

2.10 Page Programming

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

2.11 Quad Page Program (QPP 32h or 38h, or 4QPP 34h)[1]

QPP requires programming to be done one full page at a time. While less than a full page of data may be loaded for programming, the entire page is considered programmed, any locations not filled with data are left as ones, the same page must not be programmed more than once; thus, ECC is also programmed by the QPP operation.

Writing more than once to a 16-byte Program Block after the prior erase can disable ECC on that block. It is recommended to write only once to any given Program Block, then erase and repeat as necessary

2.12 OTP Program (OTPP 42h)

Programming more than once within an ECC unit disables ECC on that unit.

2.13 Command Summary by Instruction

2.13.1 FL-S Command Summary by Instruction

Table 4. FL-S Family Instruction Set — Sorted by Instruction

Instruction	Command Name	Command Description	Maximum Frequency (MHz)
18	ECCRD	ECC Read	133

2.13.2 FS-S Command Summary by Instruction

Table 5. FS-S Family Instruction Set - Sorted by Instruction

Instruction	Command Name	Command Description	Maximum Frequency (MHz)	Address Length (Bytes)
18	4ECCRD	ECC Read	133	4
19	ECCRD	ECC Read	133	3 or 4

2.14 Device ID and Common Flash Interface Field Definitions

Table 6. CFI Alternate Vendor-Specific Extended Query Parameter 94h ECC

Parameter Relative Byte Address Offset	Data	Description
00h	94h	Parameter ID (ECC)
01h	01h	Parameter Length (The number of following bytes in this parameter. Adding this value to the current location value +1 = the first byte of the next parameter.)
02h	10h	ECC unit size byte, FFh = ECC disabled

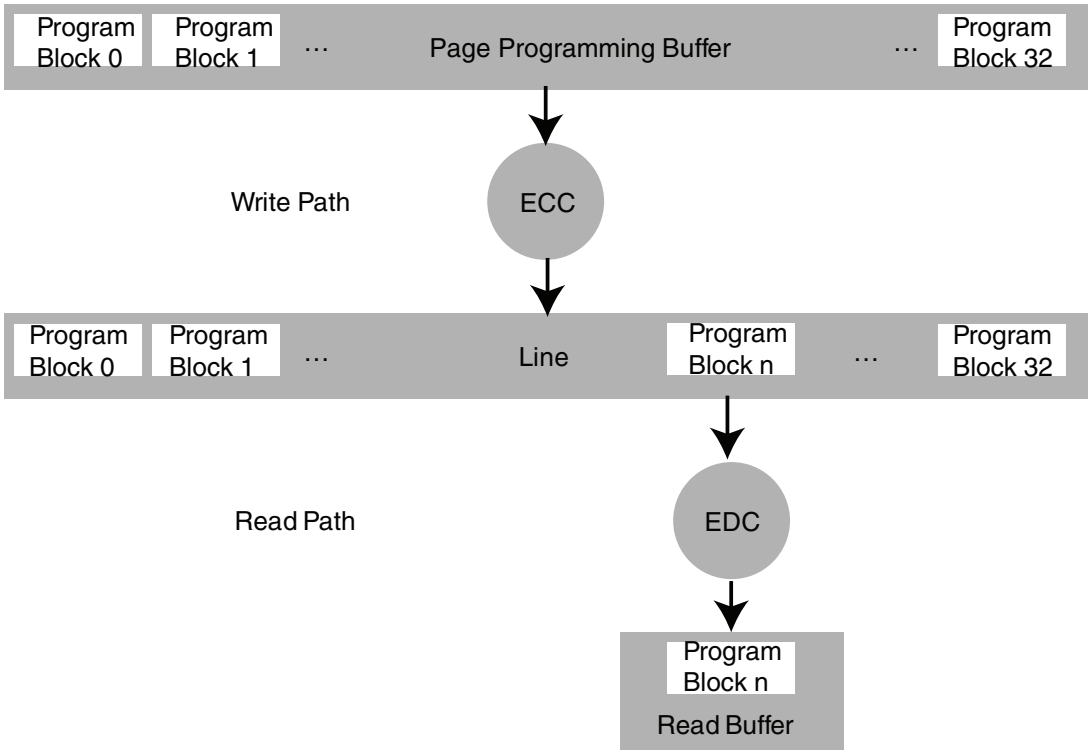
1. The FS-S family does not support the Quad Page Program (QPP or 4QPP) commands; instead use Page Program in Quad All mode.

3 Programming Guide

To better understand how the Automatic ECC feature responds to normal user operations let us first take a closer look at the relevant structures embedded within the flash chip. Figure 5 shows the 16-byte Program Block and its accompanying, but hidden, ECC information where each bit in the figure is a non-volatile flash memory bit. As shown in Figure 6 there are 32 (or 16 Program Blocks, depending upon the ordering option) Program Blocks, each with its own unique ECC information, comprising a 512-byte (or 256-byte) Line. The internal flash architecture also contains a 512-byte (or 256-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. Similarly during Flash Memory Array read operations, a separate Program Block-sized Read Buffer is used to temporarily hold information that is read from a Program Block in the Flash Memory Array and passed to the user. The Automatic ECC logic may affect data transfers from the Page Programming buffer to the Line or from the Program Block to the Read Buffer, and is enabled or disabled based on the type and sequence of operations that have been applied to each Program Block.

Figure 5. Sixteen-Byte Program Block and Associated ECC Information that Comprise an ECC Unit

Figure 6. Logic Diagram for Read and Write Paths with Automatic ECC



Let us now look at how normal flash operations affect the structures described in the previous paragraph with particular emphasis on the ECC information and the activity of the ECC and EDC logic blocks.

The Sector Erase command has the effect of changing each non-volatile flash bit in the Sector to the 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, so when a Flash Memory Array read request triggers the initial load of an erased Program 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 user-supplied bytes from the Page Program buffer to the appropriate Line. Continuing the discussion for a single

Program Block taken from the set of Program Blocks supplied by the user, the program command copies the user-supplied bytes from the Page Programming buffer to the corresponding Program Block by changing 1 bits to 0 bits as necessary. ECC parity information is computed for this Program Block and is stored in the associated ECC information area together with an indication that EDC is enabled for this Program 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 Program Block to the Read Buffer, EDC may be applied to correct at most one bit in either the Program Block or its ECC information.

Continuing from the non-erased state created by the program operation described in the previous paragraph, if a subsequent programming operation of any type is applied to the Program Block under consideration then the user data is copied from the Page Program buffer to the Program 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 since a program command cannot perform this transition. Incremental ECC parity is computed and if the new ECC parity would require any forbidden 0 bit to 1 bit transitions, the ECC information is updated to disable EDC logic for that entire Program Block; otherwise if only allowed 1 bit to 0 bit transitions are required, then the incremental parity bits are programmed and ECC remains enabled for that Program Block. For any Program Block where EDC is disabled, EDC remains disabled for that Program Block 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 Program Block to the Read Buffer, EDC is not applied.

It should be clear from the prior discussion that the new 65-nm FL and FS families of devices are backwards compatible with all of your existing flash applications; that is, the new Automatic ECC feature is transparent to the ways in which you use flash memory today. Furthermore, it should be clear that should your application demand it, the simple rule for enabling Automatic ECC for an erased Sector is to always write only once to any given Program Block; then erase and repeat as necessary.

The following sections address the two methods recommended by Cypress to enable your application for high reliability applications. 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.

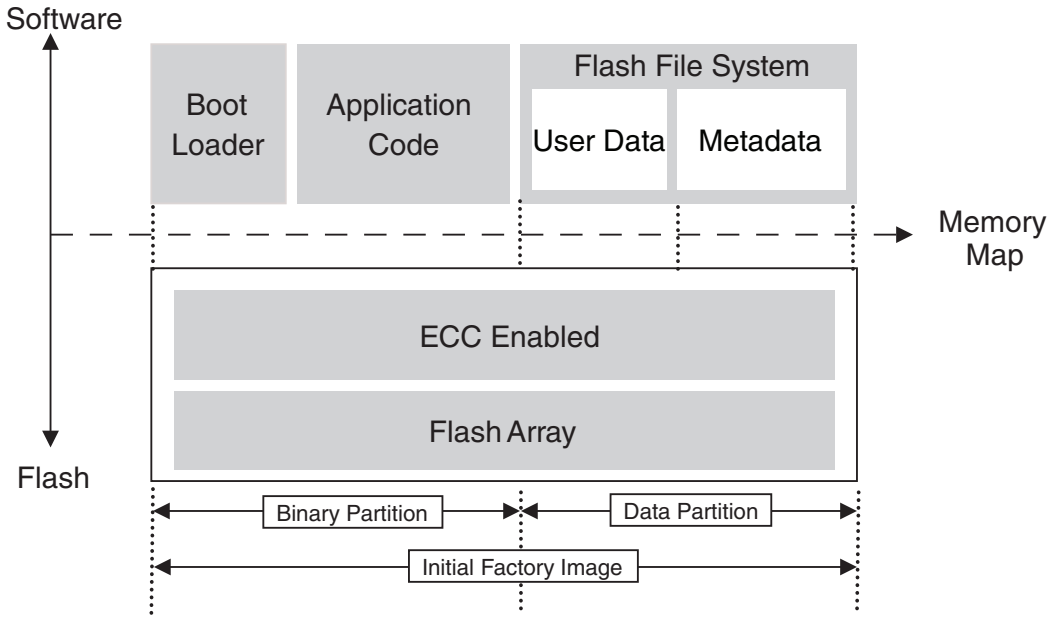
3.1 High Reliability Usage: 100% ECC Fraction

Referring to Figure 7, the best practice for factory programming the Initial Factory Image, for field reprogramming any element of the Binary and Data Partitions, and for operating applications over any element of the Binary and Data Partitions, is to program full Lines for the entire span of the programmed region, taking care to ensure that no Line is programmed more than one time after the prior erase. Not only does this practice deliver the fastest programming performance, but it also ensures that the Automatic ECC feature is enabled for every Program 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 Program Block, again taking care to ensure that no Program Block is programmed more than one time 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 Program Block, where the software ensures that no Program Block is programmed more than one time 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:

$$\text{ECC Fraction} = \frac{\#\text{ Program Blocks with ECC enabled}}{\#\text{ Program Blocks in the flash device}}$$

Figure 7. 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. Indeed 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. Cypress has prototyped this approach for one NAND file system to prove the approach. The main task for this effort is aligning all of the flash format elements with the Program Block boundaries in the 65 nm FL and FS families of flash devices. Once this is done the normal single-pass programming methods used for NAND handle the rest.

It is also possible to adapt some file systems and block drivers to achieve 100% ECC Fraction by adjusting some parameters and code to re-align the data format elements to be Program Block-aligned.

Table 7 on page 11 shows a list of data storage solutions that are known at the time of publication to support your goal of 100% ECC Fraction.

3.2 High Reliability Usage: 100% Effective ECC Fraction

As shown in Figure 8, the Binary Partition is managed with 100% ECC Fraction according to the methods described in the previous section, while 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. The Effective ECC Fraction is defined below:

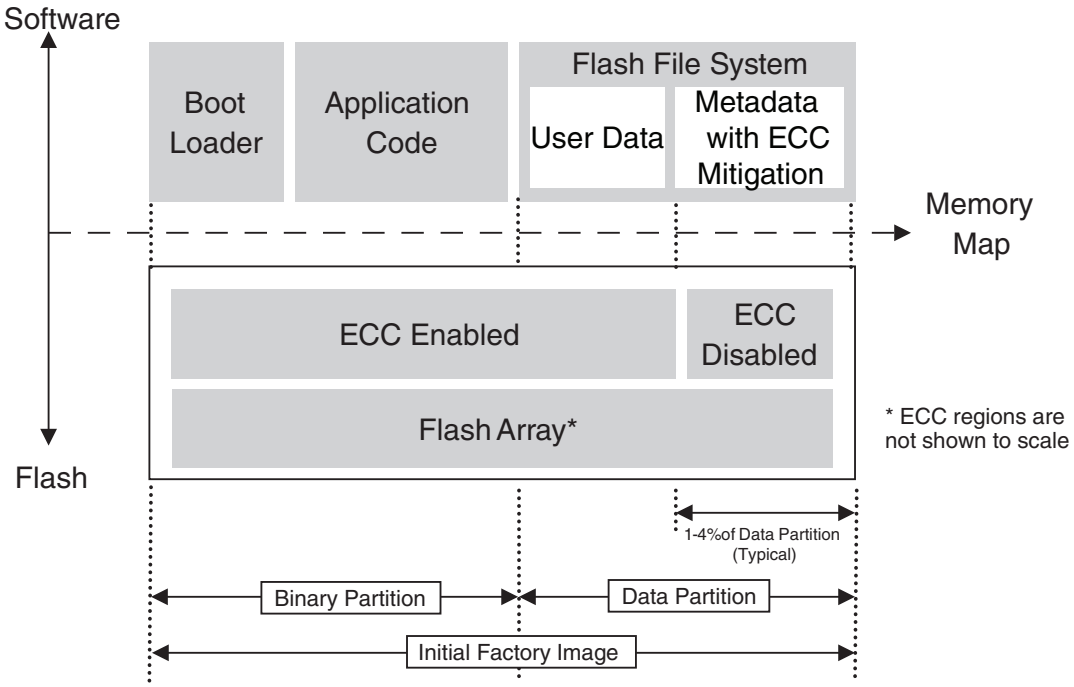
$$\text{Effective ECC Fraction} = \frac{\#\text{ Program Blocks with ECC enabled} + \text{Mitigated Program Blocks}}{\#\text{ Program Blocks in the flash device}}$$

In this definition a “Mitigated Program Block” is a Program Block where flash operations have disabled or will disable the Automatic ECC function for that Program Block, but where additional redundancy is provided by the software that effectively 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 – so this approach simply replaces the disabled Automatic ECC with alternative redundancy that is managed by the software for all sub-Program Block writes. Any redundancy method can be used so long as you are sure that the method can return correct data in the event of a single bit error within the set of bits used to represent the metadata element stored within a Program Block. This approach can be less disruptive to your trusted software stack since only a minor patch needs to be re-qualified – the core algorithms of your current NOR file system are unaffected. However, backwards compatibility with your

prior on-flash formats is affected, since 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. Please refer to Table 7 on page 11 for a list of data storage solutions that are known at the time of publication to support your goal of 100% Effective ECC Fraction.

Figure 8. Reference Memory Map for the Case of 100% Effective ECC Fraction

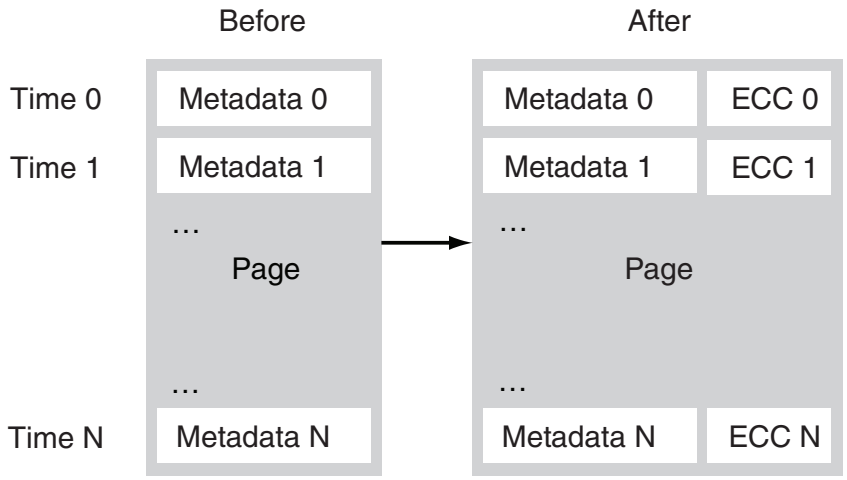


3.2.1 Example 1: Software ECC

This example is relevant for logging algorithms wherein you have a simple list of metadata elements, all smaller than a Program 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 ECC data to each metadata element, where these extra bits are written to the flash at the same time as the metadata element. Clearly your algorithm cannot overwrite these regions, since this action may invalidate the software ECC just as it might for the hardware method. Figure 9 shows the method in more detail for the case of N metadata elements stored within a single Program Block.

As the most direct approach of literally using software to replace the disabled Automatic ECC, this approach is clearly able to deliver 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. Our prototypes using this method confirm that impacts to both data storage efficiency and programming overhead can be virtually undetectable. In contrast to programming, the additional software overhead incurred by decoding software ECC on the metadata may noticeably degrade net read speed.

Figure 9. Mitigating Disabled Automatic ECC with Software ECC

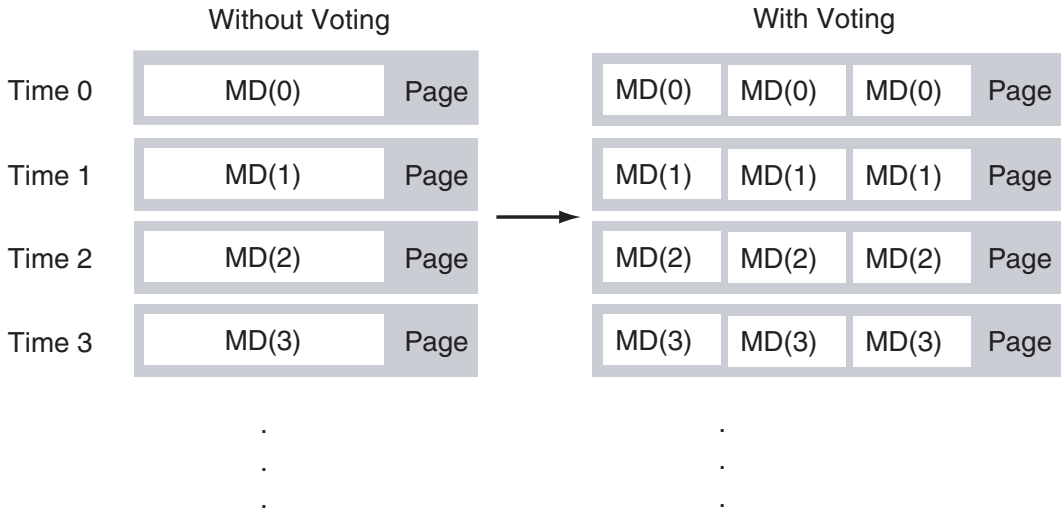


3.2.2 Example 2: Three Copy Voting

In contrast to the Software ECC method, the Three Copy Voting method is appropriate for those cases where your algorithm overwrites a metadata storage location one or more times in order 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 of the data element that are overwritten in triplicate for each update to the metadata element. Upon reading the metadata element at some later time, 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.

As implied by Figure 10, each bit of 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 is capable of returning the “true” value of that bit. When this method is applied bit-wise to the three copies of the metadata element, then one copy of every bit in the three-copy metadata element can be erroneous and the method still returns the “true” value of the metadata element. Since 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.

Figure 10. Mitigating Disabled Automatic ECC with Three Copy Voting



When the copy-wise decoding method is used the Three Copy Voting method is clearly capable of withstanding 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. Our prototypes using this method confirm that impacts to both data storage efficiency and programming overhead can be virtually undetectable.

Table 7. Data Storage Options as of Publication

Supplier	File System	< 100% ECC Fraction	100% Effective ECC Fraction	100% ECC Fraction
Cypress	SSP (65 nm)	—	—	—
Cypress	FFS – NOR/SPI	—	—	Available Now
Linux	JFFS2 UBIFS	Available Now	—	Feasible; Upon Request (1)
QNX	FFSv3	—	—	—
QNX	ETFS	—	—	—
Cypress	FMD (WinCE5, 6, 7)	—	—	—
Cypress	Flash PDD (WinCE 6 Rev 2, 7)	—	—	Available Now
Datalight	FlashFX Family	—	—	—
Blunk Microsystems	TargetFFS-NOR	—	—	—
Blunk Microsystems	TargetFTL-NDM	—	—	—

Notes:

1. *Check with your local Cypress sales representative for availability.*

4 Conclusion

The 65 nm FL and FS families of flash devices from Cypress now 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 65 nm flash without any special software considerations. For your Automotive or high reliability application, this application note also shows how you can adapt your application to achieve either 100% ECC Fraction, or 100% Effective ECC Fraction.

5 Related Documents

The following products and datasheets are related to this application note:

- S25FL127S - 128 Mbit 3.0V SPI Flash Memory Datasheet
- S25FL128S, S25FL256S - 128 Mbit and 256 Mbit 3.0V SPI Flash Memory Datasheet
- S25FL512S - 512 Mbit 3.0V SPI Flash Memory Datasheet
- S25FS064S - 64 Mbit 1.8V SPI Flash Memory Datasheet
- S25FS128S, S25FS256S - 128 Mbit and 256 Mbit 1.8V SPI Flash Memory Datasheet
- S25FS512S - 512 Mbit 1.8V SPI Flash Memory Datasheet
- S70FL01GS - 1 Gbit 3.0V SPI Flash Memory Datasheet
- S70FS01GS - 1 Gbit 1.8V SPI Flash Memory Datasheet
- S79FL256S, S79FL512S - 256 Mbit and 512 Mbit 3.0V Dual-Quad SPI Flash Memory Datasheet
- S79FL01GS - 1 Gbit 3.0V Dual-Quad SPI Flash Memory Datasheet

Document History Page

Document Title: AN200731 - Automatic ECC for Cypress 65-nm MirrorBit® Eclipse™ SPI Flash Nonvolatile Memory Families Document Number: 002-00731				
Rev.	**ECN No.**	**Orig. of Change**	**Submission Date**	**Description of Change**
**	–	–	04/06/2012	Initial version
*A	–	–	07/18/2012	Updated section: Automatic ECC Programming Guide: Logic Diagram for Read and Write Paths with Automatic ECC figure: changed Block 15 to Block 32
*B	5043065	MSWI	12/09/2015	Updated in Cypress template
*C	5328988	LIZB	06/29/2016	Updated the AN to support FS-S in addition to FL-S. Updated template
*D	5841802	AESATMP8	08/02/2017	Updated logo and Copyright.

Worldwide Sales and Design Support

Cypress maintains a worldwide network of offices, solution centers, manufacturer's representatives, and distributors. To find the office closest to you, visit us at Cypress Locations.

Products

ARM® Cortex® Microcontrollers	cypress.com/arm
Automotive	cypress.com/automotive
Clocks & Buffers	cypress.com/clocks
Interface	cypress.com/interface
Internet of Things	cypress.com/iot
Memory	cypress.com/memory
Microcontrollers	cypress.com/mcu
PSoC	cypress.com/psoc
Power Management ICs	cypress.com/pmic
Touch Sensing	cypress.com/touch
USB Controllers	cypress.com/usb
Wireless Connectivity	cypress.com/wireless

PSoC® Solutions

PSoC 1 | PSoC 3 | PSoC 4 | PSoC 5LP | PSoC 6

Cypress Developer Community

Forums | WICED IOT Forums | Projects | Video | Blogs | Training | Components

Technical Support

cypress.com/support

Cypress Semiconductor
198 Champion Court
San Jose, CA 95134-1709