

How to use error correction code (ECC) management for internal memories protection on STM32 MCUs

Introduction

This application note describes the error correction code (ECC) management and implementation on STM32 microcontrollers. This document presents both the hardware and software aspects linked to the ECC mechanism used to protect the content of internal memories. ECC protection with external memories is possible, however, its implementation is out of the scope of this document.

This document presents: general information about ECC protection, detailed hardware ECC fault management, and details of how ECC is implemented, with special focus on high performance microcontrollers. This document proposes a specific implementation of the software part of the safety solution.

This document complements the [Reference documents](#), available on www.st.com.

Table 1. Applicable products

Type	Products
Microcontroller	STM32H5 series, STM32H7 series, STM32U3 series, STM32U5 series, STM32C5 series

1 General information

This document applies to STM32H5, STM32H7, STM32C5, STM32U3, and STM32U5 series microcontrollers, which are Arm®-based devices.



Note:

Arm and Cortex are registered trademarks of Arm Limited (or its subsidiaries or affiliates) in the US and/or elsewhere.

The Arm word and logo are trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved.

Table 2. Acronyms and terms

Acronym	Definition
ECC	Error correction code
CPU	Central processing unit (part of the MCU)
CRC	Cyclic redundancy check
DED	Double-error detection
DTCM	Data-tightly coupled memory
FAR	Falling address register
ISR	Interrupt service routine
ITCM	Instruction tightly coupled memory
MCU	Microcontroller unit
MDMA	Master direct-memory access
POR	Power-on reset
RAM	Random access memory
SEC	Single-error correction
SRAM	Static RAM

Reference documents

- [1] Reference manual *STM32H503 Arm®-based 32-bit MCUs* (RM0492)
- [2] Reference manual *STM32H563/H573 and STM32H562 Arm®-based 32-bit MCUs* (RM0481)
- [3] Reference manual *STM32H7A3/7B3 and STM32H7B0 value line advanced Arm®-based 32-bit MCUs* (RM0455)
- [4] Reference manual *STM32H745/755 and STM32H747/757 advanced Arm®-based 32-bit MCUs* (RM0399)
- [5] Reference manual *STM32H723/733, STM32H725/735, and STM32H730 value line advanced Arm®-based 32-bit MCUs* (RM0468)
- [6] Reference manual *STM32U5 series Arm®-based 32-bit MCUs* (RM0456)
- [7] User manual *STM32H7 dual-core safety manual* (UM2840)
- [8] Reference manual *STM32U3 series advanced Arm®-based 32-bit MCUs* (RM0487)

2 ECC overview

The mathematician Richard Hamming invented the first ECC. The original Hamming code uses 7 bits to store 4 bits of information with redundancy bits used for correction, and detection of errors. In STM32 devices, selected RAM and flash memories are protected using a SEC-DED algorithm based on Hamming principles, but improved with one extra parity bit. The ECC code is capable of detecting and correcting a single-bit error and of detecting a two-bit error in the stored word of data.

In SRAM volatile memory, a stray alpha particle may cause a bit value to flip either way. This is a constant threat and the probability of single-bit failure is the same regardless the age of the hardware. A single-bit or two-bit error failure is a problem especially in applications where large amount of data is stored for long periods of time without reset, for example a battery-powered data logger.

In flash memory, the data decays over time, especially at high temperatures. Storage temperatures have an impact on flash memory data, but cycling (programming) temperature has an even stronger impact. The flash memory can sustain only a certain amount of rewrites to each memory word, this leads to the need of implementation of wear leveling in case of data storage. The typical retention time and life cycle of flash memory for a given product is published in the product datasheet.

Both types of failures (single-bit error and two-bit error), are inevitable, but the correct use of the ECC can prevent data loss.

Table 3. Number of extra check bits used for SEC-DED

Data word width	Number of redundancy bits
16	6
32	7
64	8
128	9
256	10

2.1 ECC implications

ECC is a key element in embedded systems that aim to comply with requirements of safety standards such as IEC60730 Class C or IEC 61508 SIL2 and higher.

A system without hardware ECC may still meet target safety standards compliance but it requires a deployment of considerable software overhead. The use of ECC memory to increase the overall diagnostic coverage above 90% is easy to do and increases the system's readiness to comply with high safety standards. Another advantage of using ECC is the potential improvement of security as ECC usage may lead to detection of hardware tampering.

2.2 RAM ECC

The RAM ECC functionality of the STM32 devices has a peripheral-like interface with registers for settings and with interrupts that permit a quick reaction to detected fault. All STM32H72x/H73x/H74x, and STM32H75x SRAM and instruction/data cache memories are protected with ECC. The data width is 64-bit for AXI-SRAM and for ITCM-RAM. All other volatile memories are accessed by 32-bit bus width (word size). On STM32H7Ax/H7Bx, only the tightly coupled memories and instruction/data cache memories are protected with ECC. The other SRAMs are not protected with ECC.

The ECC Monitoring is local per RAM region and each region is controlled by the RAMECC_MxCR. The RAMECC_IER allows a global ECC monitoring of all the RAM sectors when activated whatever the configuration of RAMECC_MxCR

Other series featuring the RAM ECC protection are STM32H5, STM32C5, and STM32U5. Based on Cortex®-M33, these series have a simpler memory architecture. They have ECC implemented on SRAM regions intended for data, which are backup SRAM, SRAM2, and SRAM3. In case of SRAM3, the use of ECC is limited to the first 256 Kbyte of SRAM3 memory range. Last 64 Kbyte block is then made unavailable for the user, serving for ECC redundancy storage.

Table 4. SRAM ECC coverage in selected STM32 devices

-	H72x/H73x	H74x/H75x	H7Ax/H7Bx	H5	U5	H7Rx/H7Sx	C5
AXI	64b	64b	X	-	-	64b ⁽¹⁾	-
ITCM	64b	64b	64b	-	-	64b	-
DTCM	32b	32b	32b	-	-	32b	-
SRAM1	32b	32b	X	X	X	32b	X
SRAM2	32b	32b	X	32b	32b	X	32b
SRAM3	-	32b	-	32b ⁽²⁾	32b ⁽³⁾	32b	-
SRAM4	32b	32b	-	-	X	X ⁽⁴⁾	-
Backup RAM	32b	32b	X	32b	32b	32b	-

1. AXI bus is 64b, but the same memory can also be accessed by AHB.
2. Only the first 256 KB of SRAM3; the last block of memory is used to manage the ECC.
3. Not the whole SRAM3, refer to document [6].
4. Used to provide ECC memory for SRAM1 and SRAM3.

On most of the STM32H7 series microcontrollers, the RAM ECC cannot be turned off (except STM32H7Rx/H7Sx). The ECC is powered and clocked along with the RAM and it is an integral part of the RAM interface. For example, backup SRAM can be disabled. This also disables the RAM ECC controller associated with it.

The ECC is computed on data word. If data smaller than a word is written in the volatile memory, the modification is done on a read-modify-write basis. On an incomplete access, the ECC does not write the value immediately. As it may be the very next byte or half word, it waits for the next write access. This is especially important in applications dealing with the backup SRAM. For example when modifying an array of characters, this has a very positive effect on device power consumption. However, a write operation is not completed in case of reset (the memory contents is retained without the last incomplete word write).

The method to make sure the array is always updated entirely is to write a dummy incomplete word write after each regular one. The dummy write address must be within the same memory (backup SRAM in this case).

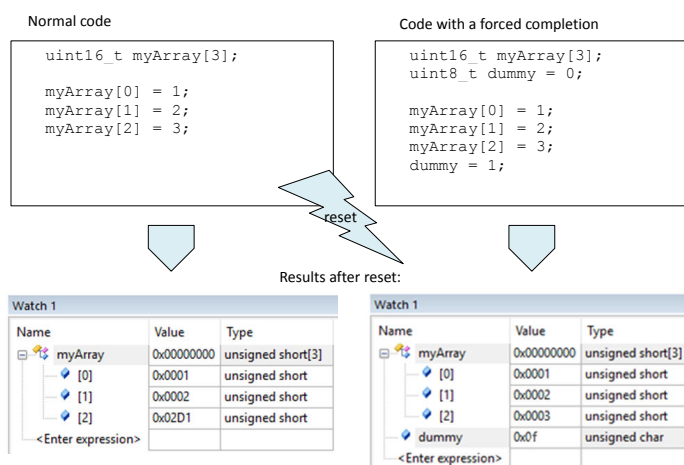
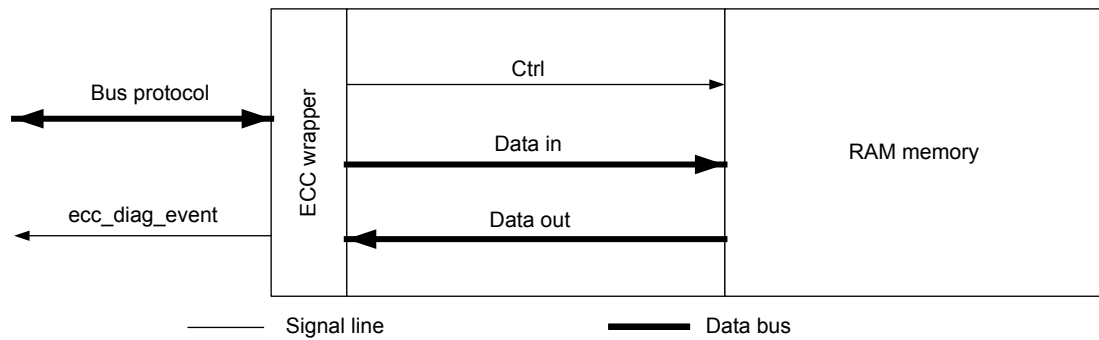
Figure 1. Unaligned access handling in preserved SRAM


Figure 2. RAM ECC controller interfaced with memory unit

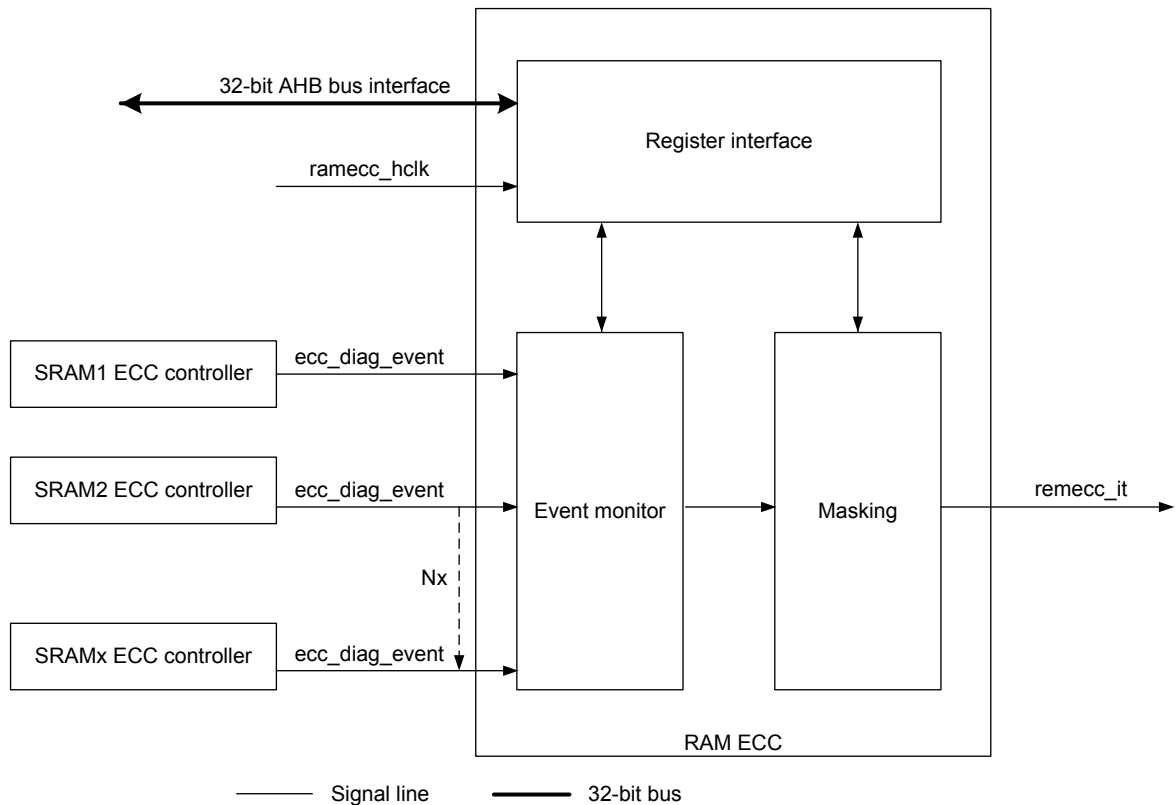


DT62937V1

RAM ECC controllers are assigned to each internal SRAM block. Within the STM32H7 architecture, the controllers are usually divided among the three system domains: D1, D2, and D3. Diagnostics from all internal SRAM units/controllers are gathered into a global control block. This global control block has a set of configuration registers and a global interrupt signal with the possibility of event masking. The STM32H7Rx/H7Sx lines are exceptions. They do not share the power domain segmented SRAM. Instead, almost all the SRAM can be accessed on the AXI bus. Parts of SRAM can be relocated between the TCM and AXI mapping. This is the only STM32H7 line where the ECC can be deactivated. This allows the area used for redundancy purposes to be available as the SRAM4.

The STM32U5 series and STM32H5 series microcontrollers have the ECC configurable for each SRAM block individually. Their ECC can also be enabled or disabled by the control bits in the OB space. Direct control of the volatile ECCE (enable) settings in RAMCFG registers is possible, but actually intended for testing rather than use by a fielded application. The STM32C5 series follows the same principle, however the ECC is only available for SRAM2.

Figure 3. ECC RAM simplified block diagram



DT62938V1

The particular RAM ECC controller assigned to a specific SRAM block checks the data integrity at each read access to that SRAM block. Some read access types are not obvious, as certain write include an implicit read phase. An example of not-obvious read access is an incomplete RAM write performed as a read/modify/write in two cycles. This can be either write of data smaller than RAM word, or an unaligned write.

2.3 Nonvolatile memory ECC

For STM32H72x/3x/4x/5x lines, the nonvolatile memory word (smallest programmable amount of memory) is 256 bits, while on STM32H7Ax/Bx lines, STM32H5 series, STM32C5 series, and STM32U5 series, it is 128 bits. Table 5 shows the word size for STM32 devices with ECC in nonvolatile memory. This is also the portion of memory protected by the bits required to achieve error detection and correction on the nonvolatile memory word. Write access to any smaller unit of memory is only possible on read-modify-write basis, this action results in higher stress to the memory hardware. The STM32H5 series, STM32C5 series, STM32U5 series, and STM32H7Ax/Bx line also feature added robustness on the small 0.5-2 Kbyte OTP area, where every 16 bits are protected with 6 bits of ECC redundancy. There is no OTP memory area on STM32H72x/3x/4x/5x lines.

Table 5. Nonvolatile memory ECC coverage in STM32 devices

Memory	H72x/ H73x	Other H74x/ H75x	H7Ax/ H7Bx	G0	G4/L4/L5/ WB/WL/U3	H503	H563/573 C5	L0/ L1 ⁽¹⁾	U5	U0
Bank1	256b	256b	128b	64b	64b ⁽²⁾	128b	128b	16b	128b	64b
Bank2	-	256b	128b	64b ⁽³⁾	64b ⁽⁴⁾	-	128b	16b ⁽⁴⁾	128b	-
Special region	-	-	16b	-	-	-	16b ⁽⁵⁾	-	-	-

1. ECC in STM32L0 has no user interface, so it is not described in detail in this table.
2. If DBANK = 0, the access width is 128 bits, but ECC remains 2 × 8 bytes.
3. Only G0Bx/G0Cx.
4. Only on dual bank devices.
5. Area of 16 bits words is configurable.

The ECC functionality is integrated into the nonvolatile memory controller and cannot be disabled. If the application is not designed to take advantage of the ECC, it can disable the associated interrupt and ignore the flag bits in the nonvolatile memory status registers.

The disadvantage of an integrated ECC solution is that programming single bits in the nonvolatile memory word is not possible without prior erase of this word. As programming without erase is sometimes used by implementations of EEPROM emulation or a monotonic counter, another algorithm must be selected for such applications on ECC enabled memories.

The flash memory controller of STM32H7 series implements also a hardware CRC integrity protection. The CRC is a complementary mechanism, not an ECC replacement. If the automated background CRC check is activated, the read access to the flash memory also implicitly checks the ECC in the whole range.

2.4 Cache memory ECC

Cache is a memory that does not have its own address range. Its purpose is to reduce the latency of accessing the addressed memory by preserving a copy of frequently accessed contents (either code or data) or contents which are likely to be needed soon (current address+1 for example). Physically it is an SRAM with different addressing.

Only STM32H7 series have cache with ECC protection. By default, the Cortex[®]-M7 L1 cache is also protected by ECC by using the same SEC DED code. The word width is 256-bit as the entire line of cache is covered. The cache ECC protection can be disabled. To modify the state of the cache ECC, the code must first disable and flush the cache. Then the ECC settings can be modified and the cache re-enabled with new setting.

The CPU cache is only involved in AXIM bus accesses, the ITCM, and DTCM address range does not require cache. The tightly coupled memories are almost exclusively dedicated to the use of the Cortex®-M7 core. The Cortex®-M7 processor can automatically recover from any detected ECC fault in instruction cache. One-bit errors are covered by the automatic correction. For two-bit errors, the line is invalidated and the instructions are loaded again from the program memory. In the case of data cache, a two-bit error detection may result in losing the ongoing modifications while reloading old data.

Note: The write through practice is not recommended as a countermeasure to this rare event.

The ART accelerator cache that supports the Cortex®-M4 core on dual-core STM32H7 series microcontrollers is not protected by ECC.

As there is no interrupt notification or interface to ECC detection results of the CPU cache, the only solution to detect the rare event of two-bit data cache ECC error is to monitor the IEBR and DEBR core registers. Refer to the document [7] and the Arm® Cortex® technical documentation (TRM) for more details.

It is also necessary to flush or invalidate instruction cache after two-bit ECC error interrupt from instruction memory (by flash or SRAM). Otherwise, illegal opcode may remain in cache, leading to hard fault.

3 ECC testing

ECC testing does not verify that the STM32 MCU is functional. It is useful to test the application firmware reaction to ECC error occurrence, therefore there is a requirement to integrate ECC testing into different aspects of general software testing. The tests can include unitary tests of ECC subsystem functions, acceptance tests, or nonregression tests. The reaction to ECC detection or correction is important, for example, for certification on safety requirements compliance.

Note: If ECC testing is related to functional safety compliance needs, it is recommended to refer to the Safety Manual of the applicable STM32 MCU series, available on www.st.com.

Note: On specific STM32 series, a dedicated software-based procedure to test ECC efficiency is available. The test procedure allows the application software to inject ECC errors in memory. Refer to the Reference Manual to check if the option is available and related details.

3.1 ECC testing in SRAM

Testing the ECC in SRAM is relatively simple because reading uninitialized RAM on a cold boot is usually enough to trigger an ECC error and test the reaction. The effect is random, but it is easy to produce.

Sometimes, for example on the STM32H7 series, it is possible to disable the ECC, modify the memory contents, and then reenable the ECC, which becomes desynced with the data by this action. Subsequent read from that memory location encounters an ECC error.

3.2 ECC testing in nonvolatile memory

Different methods can be used to test ECC, in addition to flash memory testing, as shown in [Table 6](#).

Table 6. Methods for ECC testing in nonvolatile memory

Method	Advantages	Disadvantages	Applicable products
Dedicated flash word with errors planted from factory	<ul style="list-style-type: none"> Simple to use, only requires read from the dedicated address to get the ECC interrupt Simple section of SEC or DED outcome 	<ul style="list-style-type: none"> Address decoding test is not possible Feature is only available in selected series 	STM32U5 series
Data programmed to a location that was not erased before	<ul style="list-style-type: none"> Can be applied anywhere in nonvolatile memory Known patterns to select SEC or DED outcome 	<ul style="list-style-type: none"> Feature is only available in selected series 	STM32C5 series STM32H5 series STM32H7 series
Disturbing the flash erase or program cycle with a reset	<ul style="list-style-type: none"> Universally applicable Can be applied anywhere in memory 	<ul style="list-style-type: none"> Tuning is required to provide the correctly timed reset Achieving SEC is difficult 	All STM32 series
Memory reconfiguration	Aimed at testing data area ECC protection	<ul style="list-style-type: none"> Limited to the last few sectors of nonvolatile memory Not usable once the option bytes are closed Achieving SEC is difficult 	STM32H5 STM32C5

3.3 ECC testing in flash memory

For details on dedicated memory methods, refer to the product Reference Manual.

The double write method requires writing sequences of preselected data, matching the length the flash memory word to an empty flash memory address. The memory must not be erased between writes. Reading after the second write triggers the ECC interrupt.

[Appendix A: ECC sequences](#) lists sequences that lead to SEC and DED errors for different flash memory word lengths.

The disturbed write method is sensitive to accurate synchronization between the reset and programming operations. It is necessary to execute from SRAM, or at least from a different flash memory bank. Start programming, and then, after a delay of a few microseconds, force a reset. Reading the written address after the reset should trigger the ECC error. It is easy to generate multiple bit errors this way, while consistently reproducing single-bit errors is difficult. The recommended pattern is to flip a single bit in the written data to 0 and time the reset so that the bit is written and the ECC is not updated.

Some products have the EDATA option, where certain flash memory pages can be reconfigured from code (8-Kbyte page, 9-bit ECC) to data (6-Kbyte page, 6-bit ECC). A page reconfigured to 6-bit ECC starts with ECC errors before the first erase because the reconfiguration scrambles the memory contents.

It is possible to keep the memory location programmed with user-created incorrect ECC to reproduce the error as needed. The error is encountered on each read access.

4 ECC use in applications

In order to correctly use the ECC capabilities, basic routines to deal with detected errors immediately must be implemented in the firmware. It is recommended to log and monitor the error presence for maintenance, failure prediction and hazard warning. This recommendation is especially important for safety and industrial applications.

4.1 Dealing with ECC errors in RAM

Static volatile memory is based on a symmetric arrangement of unipolar transistors. The unipolar transistors flip between two states that represent a logical 0 or 1. The amount of energy necessary to make this transition is low, hence the device keeps a low-power consumption.

A stray alpha particle may cause that a bit in the RAM changes its stored value. If the ECC mechanism is not used properly, these rare errors may accumulate over time and cause a data damage or even a system failure. These events are random by nature and occurrence of error on some address does not provide any indication where or when the next error may occur.

4.1.1 Initialization

When ECC is being used in the RAM, all memories that are to be accessed by the code must be initialized. A read access or an unaligned write to uninitialized memories is likely to trigger the ECC error due to the random initial setting of both the stored value and the redundant bits.

Any pattern is fine for the memory initialization. Find below a proposed list of steps to follow:

- Step 1.** Proceed with RAM initialization after POR or after wakeup from Standby mode or after domain standby.
- Step 2.** Clear the RAM ECC status register flags after RAM initialization.
- Step 3.** Activate the ECC error latching. Even if optional, this action is important for subsequent correction of errors and for reliability errors.
- Step 4.** Enable the interrupts for error correction and detection.
It is possible to selectively enable interrupts only for some memory regions by using register flags for particular RAM ECC controller units.
- Step 5.** Enable the global RAM ECC interrupts.

4.1.2 ECC ISR

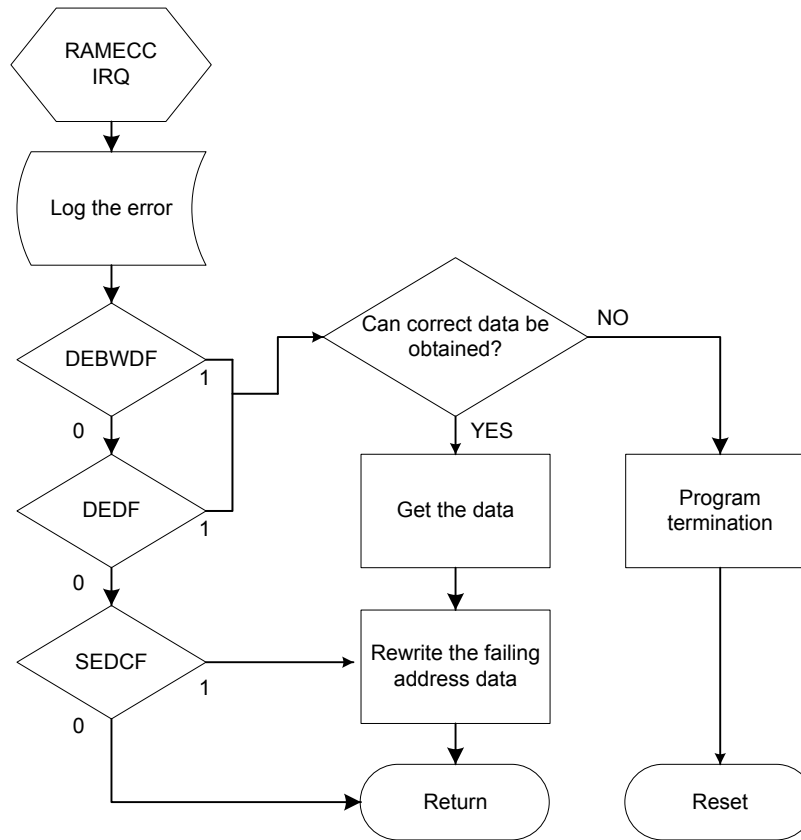
The interrupt service routine provides an opportunity to immediately react to an event of ECC error. The ISR implemented in CubeHAL generated in projects using STM32CubeMX tool is however just a start. The HAL ISR branches to a callback function. This function is not part of the HAL and this section proposes how it could be implemented.

The single-bit errors are automatically corrected by the ECC controller, but only in the data read. It is then necessary to write back the corrected data. In this case, the data and the address latching feature is very helpful. It is appropriate to write the corrected data back to its address. Failing to do so may result in a two-bit failure later (in the case of another bit within the same word is damaged).

If a two-bit error happen anyway, the subsequent action depends on what exactly was damaged. If the affected word is an instruction of code loaded to RAM, the load region within the original code in the flash memory must be identified and the code must be reloaded to the SRAM. The same kind of action is suitable for any other initialized section, for example a copy of the interrupt vector table. As a general rule, in case an instruction cache exists between the SRAM and the CPU, the cache contents must be flushed.

In case of the damaged address falling into a stack area, to avoid further damage caused by executing in incorrect context, a system reset must be performed. If the affected word address lies within boundaries of data RAM (a heap or global variables), it is up to the developer to decide which action to take. Generally a reset is recommended, but risk analysis conclusion may differ case by case.

Figure 4. RAM ECC interrupt actions example



NOTE: The abbreviations are register flags in the RAMECC monitor x status register (RAMECC_MxSR)

DT62940V1

Logging the error for subsequent analysis can be an optional part of the post-failure operation.

4.1.3 Interpreting FAR (failing address register)

The FAR (failing address register) is not showing the absolute address of the failure, but a relative location. The value in FADD[31:0] also points a word and not a byte. To compute the physical address of the failure use the following formula:

Address = Memory start address + FADD x word size in byte

For example, AXI SRAM monitor FADD=0x2004 means $0x2400\ 0000 + 0x2004 \times 8 = 0x2401\ 0020$ (64-bit words) But in SRAM1 monitor the FADD=0x2004 value interprets as $0x3000\ 0000 + 0x2004 \times 4 = 0x3000\ 8010$ (32-bit words).

A special case is DTCM and other memories, which are interleaved. To be specific, it works as follows with STM32H7 series:

- D0TCM starts at 0x20000000
- D1TCM starts at 0x20000004

In D0TCM, the address = D0TCM start address + FADD x 8 and in D1TCM address = D1TCM start address + FADD x 8. Each 64 bit of the DTCM consists of 32 bits from D0TCM followed by 32 bit of D1TCM forming common address space, but with separate ECC reporting.

4.1.4 Preventive actions for ECC in RAM

The RAM ECC events are random, hence some damage can be prevented by performing a periodic check on the used RAM area. The ECC check is activated by reading from each word; if the checkup period is appropriate, most erroneous words are detected while there is still only one wrong bit. An appropriate period can vary from hours to several days, it depends on the operating conditions, and the role of the microcontroller.

A preventive ECC checkup does not need to be completed in a single round. It is a background task that may be performed during idle moments, either by a background process or by a low-priority DMA transfers. A single loop or a DMA transfer is not possible as the SRAM is divided into non continuous address ranges.

In the case of the STM32H7 series, the MDMA is particularly suited for this task, as it can access the ITCM/ DTCM. When using the Cortex[®]-M7 CPU for memory check-by-read, the cache is involved (if cache is enabled). Accessing the SRAM through the Cortex[®]-M7 cache (so ITCM and DTCM are excluded from this rule), each read from memory fills the cache line of 256 bits. The loop that activates the ECC check on each memory word read only the first word of each 256-bit and the cache line loading continues with the remaining words. This action lowers the CPU load, but the bus remains heavily loaded.

Regular GPDMA is adequate for sweeping the memory of the STM32H5 and STM32U5 series.

Note: It is useful to track ECC incidents in memory. The monitor configuration registers in certain STM32H7 series products allow activation of failure counters.

4.2 Dealing with ECC errors in flash memory

Typical failures for the flash memory are fail due to memory cell wear and fail due to charge leakage. Some factors that might contribute to failure are interference from adjacent cell or voltage instability during programming. Unlike to SRAM, failure in certain flash memory address may indicate a slightly higher probability of a subsequent failure in the same page. Flash memory errors should be non-existent on a new device, with probability of failure increasing towards the end of projected lifetime. The flash memory lifetime depends mainly on temperature conditions and the amount of erase cycles.

4.2.1 Flash memory ECC ISR

For the flash memory of the STM32H7 series, the interrupts that notify of an ECC error are included in the flash memory global interrupt vector. The ISR checks the flash memory status register FLASH_SR1 for ECC flags “single error correction” and “double error detection” and take the appropriate action (which depends on the flash memory use). In case of ECCD, it is necessary to flush instruction cache and prevent passing corrupted opcode to the CPU. This task requires special attention, as the BusError is signaled on ECCD. Given that the interrupt vector is shared with normal flash memory operations (such as end of programming), the ISR should then pass control to HAL to deal with the other flags. On the STM32H5 series, STM32U5 series, and STM32C5 series, the ECC is signaled using a regular flash interrupt vector, while the ECCD issues a nonmaskable interrupt.

4.2.2 Flash memory code

The on-chip nonvolatile memory is primarily intended to be used for code. The code is not likely to be frequently rewritten, so if a damage occurs it is most likely caused by aging and by charge leakage. On a dual-bank device it is possible to have a second copy of the same code and swap to this second copy when an ECC error is indicated. This solution implies that the health of both bank contents is monitored. It is possible to reprogram the failing contents of the other bank from the healthy bank, however there is no guarantee on how much this action might improve the device life expectancy.

4.2.3 EEPROM emulation

If the failing Flash memory cell is used to store data, the failure cause is likely to be linked to a program/erase cycling. Advanced EEPROM emulation implementations include mechanisms that deals with failing memory cells and are able to exclude them from the cycling. The application note *EEPROM emulation techniques and software for STM32 microcontrollers*(AN4894) deals with the problematic of EEPROM emulation including the ECC implications.

4.2.4 Preventive action for ECC in flash memory

The CRC hardware module of the STM32H7 series is a useful tool to monitor the embedded flash memory health. CRC can check either the whole bank or a specific address range autonomously; ECC is implicitly checked on read as well. The program must then implement a reaction to a detected problem.

For products without this CRC feature, DMA, for example, may be used to check on rarely used portions of code. If the secure boot is in place, such periodical checkups of installed applications and loader code can be implemented here.

5 Conclusion

With a higher level of integration of the microelectronics in a system, memory cells are more prone to failure, hence ECC memory integrity protection becomes more important. The main difference between the RAM ECC compared to a regular peripheral is that the RAM ECC cannot be turned off as it is an integral part of the RAM interface.

This application note is a description of the ECC in RAM, flash memory and cache memory. It provides also procedures to deal with ECC errors in RAM and flash memory.

Appendix A ECC sequences

For ECC testing in flash memory, the following sequences that lead to SEC and DED errors for different flash memory word lengths can be used.

128-bit sets:

1. Select a flash page.
2. Erase the page.
3. Choose an address AD#1 on the page.
4. Perform a 128-bit write on AD#1 with data: 0x00000000 00000000 00000000 0000000A
5. Perform a 128-bit write on AD#1 with data: 0x00000000 00000000 00000000 0000000C

Note: This is correct, writing twice on the same address is performed on purpose.

6. Read data from AD#1 (128-bit read) and check ECC flags related to error detection/correction. Data read should be 0x00000000 00000000 00000000 0000000A and one error correction on this address must be reported.
7. Choose an address AD#2 on the page (AD#1 <> AD#2).
8. Perform a 128-bit write on AD#2 with data: 0x00000000 00000000 00000000 00000000
9. Perform a 128-bit write on AD#2 with data: 0x00000000 00000000 00000000 00000004

Note: This is correct, writing twice on the same address is performed on purpose.

10. Read data from AD#2 (128-bit read) and check ECC flags related to error detection/correction. Data read should be 0x00000000 00000000 00000000 00000000 and one error correction on this address must be reported.
11. Choose an address AD#2 on the page (AD#1 != AD#2 != AD#3).
12. Perform a 128-bit write on AD#3 with data: 0x00000000 00000000 00000000 0000000A
13. Perform a 128-bit write on AD#3 with data: 0x00000000 00000000 00000000 0000000C

Note: This is correct, writing twice on the same address is performed on purpose.

14. Perform a 128-bit write on AD#3 with data: 0x00000000 00000000 00000000 00000004

Note: This is correct, writing three times on the same address is performed on purpose.

15. Read data from AD#3 (128-bit read) and check ECC flags related to error detection/correction. Two error detections on this address must be reported.

256-bit sets:

1. Select a flash page.
2. Erase the page.
3. Choose an address AD#1 on the page.
4. Perform a 256-bit write on AD#1 with data: 0xCCCCCCCC CCCCCCCC CCCCCCCC CCCCCCCC
CCCCCCCC CCCCCCCC CCCCCCCC CCCCCCCC1
5. Perform a 256-bit write on AD#1 with data: 0xCCCCCCCC CCCCCCCC CCCCCCCC CCCCCCCC
CCCCCCCC CCCCCCCC CCCCCCCC CCCCCCCC0

Note: This is correct, writing twice on the same address is performed on purpose.

6. Read data from AD#1 and check ECC flags related to error detection/correction. Data read should be 0xCCCCCCCC CCCCCCCC CCCCCCCC CCCCCCCC CCCCCCCC CCCCCCCC CCCCCCCC
CCCCCCC1 and one error correction on this address must be reported.
7. Choose an address AD#2 on the page (AD#1 <> AD#2).
8. Perform a 256-bit write on AD#2 with data: 0xCCCCCCCC CCCCCCCC CCCCCCCC CCCCCCCC
CCCCCCCC CCCCCCCC CCCCCCCC CCCCCCBB
9. Perform a 256-bit write on AD#2 with data 0xCCCCCCCC CCCCCCCC CCCCCCCC CCCCCCCC
CCCCCCCC CCCCCCCC CCCCCCCC CCCCCCCC

Note: This is correct, writing twice on the same address is performed on purpose.

10. Read data from AD#2 and check ECC flags related to error detection/correction. Two error detections on this address must be reported.

Note: In addition to the patterns published here, there are many different effective combinations. In particular, DED is relatively simple to achieve because most rewrites tend to produce multiple bit errors.

Revision history

Table 7. Document revision history

Date	Version	Changes
27-May-2019	1	Initial release.
6-Jan-2020	2	Updated: <ul style="list-style-type: none"> Section Introduction Section 2.2 RAM ECC Section 2.3 Flash memory ECC
31-Mar-2022	3	Updated: <ul style="list-style-type: none"> Section Introduction Section 2 ECC overview Section 2.2 RAM ECC Section 2.3 Flash memory ECC Added Section 3.1.3 Interpreting FAR (failing address register)
21-Feb-2023	4	Updated document to cover STM32H5, STM32H7, and STM32U5 series microcontrollers. <ul style="list-style-type: none"> Section Introduction Section General Information Section 2 ECC overview Section 2.1 ECC implications Section 2.2 RAM ECC Section 2.3 Flash memory ECC Section 2.4 Cache memory ECC Section 3.1.2 ECC ISR Section 3.2.1 Flash memory ECC ISR Added: <ul style="list-style-type: none"> Section 2.5 ECC testing
10-Jul-2023	5	Updated: <ul style="list-style-type: none"> Section 2: ECC overview Section 2.2: RAM ECC Section 4.1.3: Interpreting FAR (failing address register)
7-Feb-2024	6	Updated: <ul style="list-style-type: none"> H7Rx/H7Sx added to Table 4. SRAM ECC coverage in selected STM32 devices H7Rx/H7Sx description added to Section 2.2: RAM ECC U0 added to Table 5. Nonvolatile memory ECC coverage in STM32 devices
11-Jun-2025	7	Updated: <ul style="list-style-type: none"> Product information to include the STM32U3 series Section Introduction Section 2.2: RAM ECC Section 2.3: Nonvolatile memory ECC
24-Feb-2026	8	Updated: <ul style="list-style-type: none"> Section 2: ECC overview Section 3: ECC testing Added: <ul style="list-style-type: none"> STM32C5 series Section 3.1: ECC testing in SRAM Section 3.2: ECC testing in nonvolatile memory Section 3.3: ECC testing in flash memory Appendix A: ECC sequences

Contents

1	General information	2
2	ECC overview	3
2.1	ECC implications	3
2.2	RAM ECC	3
2.3	Nonvolatile memory ECC	6
2.4	Cache memory ECC	6
3	ECC testing	8
3.1	ECC testing in SRAM	8
3.2	ECC testing in nonvolatile memory	8
3.3	ECC testing in flash memory	8
4	ECC use in applications	10
4.1	Dealing with ECC errors in RAM	10
4.1.1	Initialization	10
4.1.2	ECC ISR	10
4.1.3	Interpreting FAR (failing address register)	11
4.1.4	Preventive actions for ECC in RAM	11
4.2	Dealing with ECC errors in flash memory	13
4.2.1	Flash memory ECC ISR	13
4.2.2	Flash memory code	13
4.2.3	EEPROM emulation	13
4.2.4	Preventive action for ECC in flash memory	13
5	Conclusion	14
Appendix A ECC sequences		15
Revision history		16
List of tables		18
List of figures		19

List of tables

Table 1.	Applicable products	1
Table 2.	Acronyms and terms	2
Table 3.	Number of extra check bits used for SEC-DED	3
Table 4.	SRAM ECC coverage in selected STM32 devices	4
Table 5.	Nonvolatile memory ECC coverage in STM32 devices	6
Table 6.	Methods for ECC testing in nonvolatile memory	8
Table 7.	Document revision history	16

List of figures

Figure 1.	Unaligned access handling in preserved SRAM	4
Figure 2.	RAM ECC controller interfaced with memory unit	5
Figure 3.	ECC RAM simplified block diagram	5
Figure 4.	RAM ECC interrupt actions example	11

IMPORTANT NOTICE – READ CAREFULLY

STMicroelectronics NV and its subsidiaries (“ST”) reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice.

In the event of any conflict between the provisions of this document and the provisions of any contractual arrangement in force between the purchasers and ST, the provisions of such contractual arrangement shall prevail.

The purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST’s terms and conditions of sale in place at the time of order acknowledgment.

The purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of the purchasers’ products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

If the purchasers identify an ST product that meets their functional and performance requirements but that is not designated for the purchasers’ market segment, the purchasers shall contact ST for more information.

ST and the ST logo are trademarks of ST. For additional information about ST trademarks, refer to www.st.com/trademarks. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2026 STMicroelectronics – All rights reserved