AN12522

S32K1xx ECC Error Handling

Rev. 0 — July 2019 Application Note

by: NXP Semiconductors

Contents

1 Introduction

Error Correction Code, or ECC, is commonly utilized with memories in applications where data corruption via soft-errors is not easily tolerated. Software errors can be caused by radiation (a neutron or alpha particles), electro-magnetic interference, electrical noise, or a short circuit between cells.

The ECC method used in S32K1xx family, provide Single Error Correction (SEC) and Double Error Detection (DED) capability.

This application note is intended to describe how the ECC protection is implemented in S32K1xx family and understand the particular ECC event response for S32K14x and S32K11x devices. It also offers some software examples for illustration of mentioned behavior.

2 SRAM ECC error handling

2.1 SRAM initialization after power-on-reset

The SRAM is split in two regions: SRAM_L and SRAM_U. The RAM is implemented such that the SRAM_L and SRAM_U ranges from a contiguous block in the memory map. For more details please refer the S32K1xx_memory_map.xlsx file attached in the Reference Manual.

NOTE

The SRAM_L is not ECC protected for the S32K11x devices. The FlexRAM used as system RAM is not ECC projected. LPUART and FlexCAN RAM are ECC protected.

The reset state of the SRAM is unknown, thus data may contain random data. Most probably the first read attempt to any address would generate non-correctable ECC error. It is essential that each memory address be written to a known value before being read. If an uninitialized memory address is read, it is likely the read will result in a multiple-bit ECC error and an errored transaction on the AHB. Therefore, the SRAM must be initialized after power-up and this includes the peripherals RAM memories as the LPUART and FlexCAN RAMs.

2.2 Used ECC algorithm

The ECC implementation for the SRAM uses a Modified Hamming Code scheme with 40-bit check base that consists of 32-bits of data plus 8-parity bits.

2.3 ECC for single bit errors

The S32K14x and S32K11x devices handle the single bit error in the same way. The Error Reporting Module (ERM) can generate an interrupt (if enabled) to notify a single-bit correction event which can be enabled setting the CR0[ESCIEx] bit to 1. There are three ways to proceed with the error after the event has happened:

1. **Normal MCU operation**: The MCU detects the fault (fault detection time) and then corrects it (fault reaction time). At this point the MCU continues operating correctly.

2. **With an appropriate functional safety mechanism to manage the fault**: The MCU detects the fault (fault detection time) and then corrects it (fault reaction time). At this point the software has to switch to a safe_state_system (a safe_state_system is an operating mode without an unreasonable probability of occurrence of physical injury or damage to the health of any persons). The safe_state_system has to be defined by the user.
3. **Without any suitable functional safety mechanism**: The MCU detects the fault (fault detection time) and then corrects it but the user was not notified that an error happened. A hazard may appear after the Fault Tolerant Time Interval (FTTI) has elapsed.

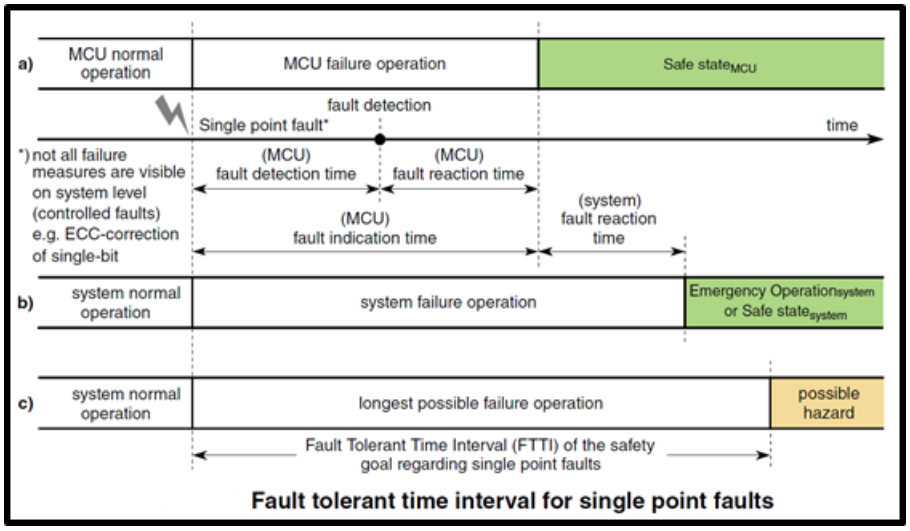

Figure 1. Fault tolerant time for single point faults

2.4 ECC for non-correctable errors

The S32K14x and S32K11x devices handle the non-correctable error in a different way. The Error Reporting Module (ERM) can generate an interrupt (if enabled) to notify a double -bit error event, which can be enabled setting the CR0[ENCIEx] bit to 1.

- **S32K14x Error Handling:** When the fault occurs, the CPU jumps to Error Reporting Module (ERM) interrupt handler (if enabled).
- **S32K11x Error Handling:** When the fault occurs, the CPU jumps to the hard fault error interrupt handler first and just after that, the ERM interrupt flag is set. At this point, the user has to check if the ERM status register is showing a non-correctable ECC error event. If yes, it is recommended to follow the recommended reaction (system reset).

NOTE

It must be ensured that no read access is done onto SRAM_U during hard fault exception handling or it will cause core lock up. Hence, linker file should be updated such that stack, vector table, variables used in hard fault handler etc. should not be in SRAM_U area. SRAM_L or FlexRAM can be used for this.

There are three ways to proceed, once this event has happened:

1. **Normal MCU operation**: The MCU detects the fault (fault detection time) and then reports it (fault reaction time).
2. **With an appropriate functional safety mechanism to manage the fault**: The MCU detects the fault (fault detection time) and then reports it (fault reaction time).
3. **Without any suitable functional safety mechanism**: The MCU detects the fault (fault detection time). A hazard will appear after the Fault Tolerant Time Interval (FTTI) has elapsed.

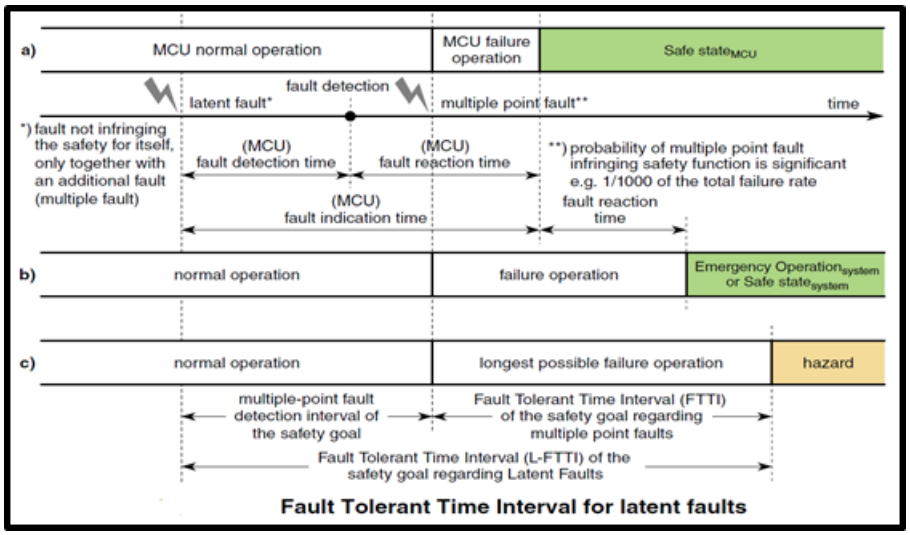

Figure 2. Fault tolerant time interval for latent faults

2.5 SRAM error injection

The Error Injection Module (EIM) is used for diagnostic purposes to induce single-bit and multi-bit inversions on read data when accessing the SRAM. When the EIM is used to inject faults on memory, any access to the SRAM generates the corresponding error injected (single or double bit error).The EIM supports two error injection channels which correspond to a specific RAM array.

The following table shows the channel assignments for the different S32K1xx derivatives.

Table 1. SRAM injection channels

EIM Channel	RAM Array	
	S32K14x	S32K11x
0	SRAM_L	SRAM_U
1	SRAM_U	Reserved

For safety applications, it is recommended to check the ECC functionality injecting errors into the SRAM to check the reporting of such errors.

3 FLASH ECC error handling

The ECC logic implemented in the S32K1xx Flash memory can correct single-bit fault automatically and can detect multiple-bit fault in each NVM sections. The multiple-bit fault is enabled using the FERCNFG[DFDIE] bit. When the multiple-bit error is detected, the FERSTAT[DFDIF] flag is set, and the interrupt request is generated.

When the FlexNVM region is configured as Emulated EEPROM, any single-bit ECC errors are automatically corrected before copying data into EEERAM at the read access from Emulated EEPROM. Any double-bit ECC errors on valid Emulated EEPROM locations which contains data that need to be copied to EEERAM and are reflected as the corresponding data records left as all 1's in EEERAM.

3.1 Used ECC algorithm

The ECC implementation for the NVM memories use a Modified Hamming Code scheme with 72-bit check base that consists of 64-bits of data plus 8-parity bits.

3.2 ECC for non-correctable errors

The S32K14x and S32K11x devices handle the non-correctable error in the same way. The Flash Memory Module (FTFC) can generate an interrupt (if enabled) to notify a multiple -bit error event which can be enabled setting the FERCNFG[DFDIE] bit to 1.

- **S32K1xx Error Handling:** When the fault occurs, the FERSTAT[DFDIF] flag is set notifying that a double bit fault was detected. The flash controller will generate a AHB error response resulting in a bus fault (if enabled). After serving the bus fault, jump to the Flash Memory Module (FTFC) interrupt handler (if enabled). The software can handle the error depending on whether the error occurred in Code Space or Data Space.

NOTE

- If an uncorrectable error fault occurs during execution of a machine exception, a safe state shall be entered.
- The bus fault is disabled by default. Therefore, it will be getting escalated to hard fault.

3.3 Flash error emulation

The Flash Memory Module (FTFC) allows the users to emulate the setting of the FERSTAT[DFDIF] flag to check the associated interrupt routine. Setting the FERCNFG[FDFD] bit creates the error emulation.

CAUTION

Consider that the real flow is first jump into the BusFault (if enabled) before going into the Flash error handler.

NOTE

Cumulative programming of bits (back-to-back program operations without an intervening erase) within a flash memory location is not allowed.

4 References

- S32K1xx Series Reference Manual by NXP semiconductors.

How To Reach Us

Home Page:

nxp.com

Web Support:

nxp.com/support

Information in this document is provided solely to enable system and software implementers to use NXP products. There are no express or implied copyright licenses granted hereunder to design or fabricate any integrated circuits based on the information in this document. NXP reserves the right to make changes without further notice to any products herein.

NXP makes no warranty, representation, or guarantee regarding the suitability of its products for any particular purpose, nor does NXP assume any liability arising out of the application or use of any product or circuit, and specifically disclaims any and all liability, including without limitation consequential or incidental damages. “Typical” parameters that may be provided in NXP data sheets and/or specifications can and do vary in different applications, and actual performance may vary over time. All operating parameters, including “typicals,” must be validated for each customer application by customer's technical experts. NXP does not convey any license under its patent rights nor the rights of others. NXP sells products pursuant to standard terms and conditions of sale, which can be found at the following address: nxp.com/SalesTermsandConditions.

While NXP has implemented advanced security features, all products may be subject to unidentified vulnerabilities. Customers are responsible for the design and operation of their applications and products to reduce the effect of these vulnerabilities on customer’s applications and products, and NXP accepts no liability for any vulnerability that is discovered. Customers should implement appropriate design and operating safeguards to minimize the risks associated with their applications and products.

For more information, please visit: http://www.nxp.com

For sales office addresses, please send an email to: salesaddresses@nxp.com

Date of release: July 2019

Document identifier: AN12522

arm