NXP Semiconductors
Application Notes

Document Number: ANxx
Rev. 0 Draft A, 09/2021

Error Correction Codes Implemented on MPC5777C or MPC5775B/E

1. Introduction

The Error Correction Code (ECC) is commonly utilized with memories in applications where data corruption via soft errors (SEU) is not easily tolerated.

Soft errors can be caused by radiation, electro-magnetic interference, or electrical noise.

The intention of this application note is to describe how the ECC protection is implemented with the MPC5777C or MPC5775B/E device and to understand the MCU's ECC event response.

Because MPC5777C or MPC5775B/E implements the Error Detection Code (EDC) in certain places, this document marginally mentions this topic as well.

This document also slightly compares the approaches used across the MPC57xx family.

NOTE

All diagrams are simplified and may not contain all implementation details. It may omit technical details not important for the purpose of this document.

NOTE

Provided information is related to MPC5777C and MPC5775B/E if it is not stated otherwise.

Contents

2. ECC protected memory initialization

2.1. SRAM initialization after power-on-reset

The reset state of the internal SRAM is random so the data and checkbits may contain any data. Most probably, the first read attempt to any address generates a non-correctable ECC error. The SRAM must be initialized after a powerup. It means that the whole SRAM is deleted or written by any value. A 64-bit write is needed to completely define the ECC code for the data unit.

Smaller write accesses (32-bit/16-bit/8-bit) cause the read-modify-write operation with ECC error-affected data.

```
# Store number of 128Byte (32GPRs) segments in Counter
e_lis   r5, _SRAM_SIZE@h   # Initialize r5 to size of SRAM (Bytes)
e_or2i  r5, _SRAM_SIZE@l
e_srwi  r5, r5, 0x7        # Divide SRAM size by 128
mtctr   r5                 # Move to counter for use with "bdnz"
# Base Address of the internal SRAM
e_lis   r5, _SRAM_BASE_ADDR@h
e_or2i  r5, _SRAM_BASE_ADDR@l
# Fill SRAM with writes of 32GPRs
sram_loop:
e_stmw  r0,0(r5)            # Write all 32 registers to SRAM
e_addi  r5,r5,128           # Increment the RAM pointer to next 128bytes
e_bdnz  sram_loop           # Loop for all of SRAM
```

2.2. SRAM initialization after reset (preserving STANDBY data)

2.3. Initialization of other embedded SRAM memories

2.3.1. eDMA RAM arrays

Automatically:

The initialization of the eDMA Transfer Control Descriptor (TCD) memory is performed by the eDMA controller itself after a reset.

The initialization runs for 256 eDMA clock cycles. All fields in the TCD memory are initialized to 0. All application read or write accesses to the TCD are delayed until the initialization is finished.

2.3.2. FlexCAN RAM array

TBD

2.3.3. FEC RAM array

TBD

2.3.4. eTPU SDM and SCM RAM arrays

TBD

3. Used error detection/correction codes

3.1. Terminology used within this document

- Error Correction Code (ECC) - adding checksum bits to protected data. All employed ECC algorithms provide the SECDED capability.
- Single Error Correction/Double Error Detection (SECDED).
- 1-bit/single-bit/correctable ECC error - 1 faulty bit within the data unit or checksum that can be corrected by SECDED.
- 2-bit/multi-bit/non-correctable/uncorrectable ECC error – 2 (or more) faulty bits within the data unit or checksum that can only be detected by SECDED.
- End-to-End ECC (e2eECC) - provides an additional layer of safety by including the ECC on all bus transactions. The ECC for the transfer is generated on the transmitting end of the transaction and checked at the receiving end.
- Error Detection Code (EDC) - adding checksum bits to the protected data. The employed EDC algorithms provide Double Error Detection capability. The term EDC is used in our documents in multiple meanings. In most cases, it is additional EDC protection (supervision), checksum being added and subsequently removed from/to data, address, attribute, or other signals, according the specific needs of the protected transfer or memory module. It is further protection capable to find an ECC malfunction.
- ECC transformation (Checkbit Transformation) - internal busses may use different granularities (either 64-bit or 32-bit). On the interface of two different busses, the checkbit transformation (i.e. logical operations changing data checksum from one format to another) or the reverse transformation is necessary.
- ECC manipulation - further generalization of any ECC checkbits re-coding, either because of a less than 64-bit access or due to removing the address portion from the ECC (changing from the e2eECC on internal busses to the ECC in the target memory).

3.2. Used ECC algorithms and error responses

The employed ECC protection variants used with the MPC57xx internal memories use the Hsiao Code in 64-bit or 32-bit granularities.

3.3. e2eECC protection of transfers over system buses

MPC5777C (e200z759):

e2eECC gasket

3.4. Internal FLASH

MPC5777C: 64-bit data + 8-bit ECC

3.4.1. Code FLASH

Single-bit ECC events on code flash accesses are automatically corrected and reported to the ERM (if enabled in the flash controller by the procedure described in Intentional generation of FLASH 1b ECC error). On a multi-bit ECC event, the core responds with a bus error, as summarized in e200z7 core response on ECC event, and an error is reported to the ERM module.

3.4.2. Data FLASH

The reporting of ECC events on data flash accesses is device-specific. It can be configurable or suppressed, as shown in Table 1.

When the ECC event reporting is suppressed, single-bit and multi-bit ECC errors are not reported (either to the core or to the ERM module). On a multi-bit ECC event, the corrupted read data is replaced with a fixed, ECC-clean illegal opcode value.

When the ECC event reporting is configurable and enabled, single-bit ECC events on data flash accesses are automatically corrected and reported to the ERM (if enabled in the flash controller by the procedure described in Intentional generation of FLASH 1b ECC error). On a multi-bit ECC event, the core responds with a bus error, as summarized in e200z7 core response on ECC event, and an error is reported to the ERM module.

When the ECC event reporting is configurable and disabled, it behaves the same way as if it was permanently suppressed.

Table 1. ECC on data flash accesses

	ECC event reporting on data flash access is suppressed permanently / optionally	Returned ECC-clean illegal opcode value
MPC5744P	permanently	0xFFFF_FFFF
MPC5746C	permanently (only HSM data flash)	0x1555_1555
MPC5748G	permanently (only HSM data flash)	0x1555_1555
MPC5746R	optionally, PFCR3[DERR_SUP]	0x1555_1555
MPC5775K	permanently	0x1555_1555
MPC5777C	optionally, PFCR3[DERR_SUP]	0x1555_1555
MPC5775B/E	optionally, PFCR3[DERR_SUP]	0x1555_1555
MPC5777M	permanently	0x1555_1555

3.5. Internal SRAM

MPC5777C: 64-bit data + 8-bit ECC

Single-bit ECC events on the internal SRAM accesses are automatically corrected and reported to the ERM. On a multi-bit ECC event, the core responds with a bus error, as summarized in e200z7 core response on ECC event, and an error is reported to the ERM module.

3.6. ECC on other embedded internal SRAM memories

For a complete picture, the following MPC5777C's and MPC5775B/E's embedded RAMs are ECC-protected as well:

- eDMA RAM arrays
- FlexCAN RAM array
- FEC RAM array
- eTPU SDM and SCM RAM arrays

3.7. Used EDC protection (supervision)

3.7.1. Flash/SRAM

Additional EDC transfer protection may be located in the flash array or flash controller due to ECC manipulation (either because of less than 64-bit access ECC transformation or because of the transfer protection between the flash array and the flash controller when the ECC is re-coded due to a removal of the address portion from the transferred packet address, data, or e2eECC).

3.7.2. Crossbar Integrity Checker (XBIC)

The XBIC is sub-module verifying the integrity of the attribute information for XBAR transfers using an 8-bit Error Detection Code (EDC). The XBIC integrity checking is independent from the end-to-end ECC that covers the transfer address and data.

The EDC(72,64) code, which protects against single- and double-bit error of 64-bit attribute information (transfer direction, type, size, protection control, burst type, and so on) is used.

4. MPC5777C/MPC5775B/E ECC/EDC system implementation

TBD

5. e200z7 core response on ECC event

The correctable (single-bit) errors are automatically corrected and they can be tracked by the ERM.

The non-correctable (multi-bit) ECC error causes the Machine Check Exception (IVOR1) and a potential interrupt generated via ERM-related FCCU non-critical fault input and its ALARM state (IVOR4 for the software vector mode or vector 488 for the hardware vector mode) or ERM interrupt.

Table 2. Core reaction to detection of multiple-bit ECC error for e200z7

MSR[EE]	MSR[ME]	Access Type	Result
x	0	Instruction or data	Machine Check Interrupt (IVOR 1) Error flags in MCSR register are ignored
x	1	Instruction or data	Machine Check Interrupt (IVOR 1) Error flags in MCSR register must be cleared in exception service routine to avoid IVOR 1 recall

The ERM/FCCU configuration is further described in Interrupt handling of non-correctable ECC error.

6. Behavior in case EDC event occurs

The MPC57xx architecture offers advanced EDC transfer protection, such as the XBIC attribute EDC, RAM controller EDC, and flash controller EDC (see the device reference manual for a specific implementation).

The detection of such errors triggers the EDC-related FCCU non-critical fault input. The FCCU configuration is further described in FCCU.

7. Correctable ECC error servicing

The correctable ECC error servicing is not needed, because 1-bit ECC errors are automatically corrected during data/instruction read. The detection of 1-bit errors could sense gradual degradation of the flash memory content caused by aging. A 1-bit error may be corrected by reading the data and writing them back.

8. Non-correctable ECC error servicing

The fixing of multi-bit ECC errors is application-dependent and it must be part of the ECC error interrupt handling.

It can be based on the IVOR1 exception handler or IVOR4 interrupt handler, as described in IVOR1 exception handling of non-correctable ECC error and Interrupt handling of non-correctable ECC error.

8.1. IVOR1 exception handling of non-correctable ECC error

This document describes the IVOR1 (machine check) handling, because it is the most common one and it is compatible with all e200 cores. The precondition is to have MSR[ME]=1, because it guarantees that the exception is not only directly associated with the current instruction execution stream (synchronous exceptions), but also with those reported by the subsystem as a bus error termination (asynchronous exceptions, for example cache line filling). However, this approach still catches only errors related to the core.

If ECC non-correctable errors caused by other master than the core are supposed to be serviced, use the handling based on the ERM interrupt (Interrupt handling of non-correctable ECC error) or a combination of both approaches.

8.1.1. Machine Check Syndrome Register (MCSR)

The register provides a possibility to differentiate between the sources of machine check exceptions. The exception service routine should analyze the root cause of the exception:

- The error is caused by reading ECC-corrupted data sets MCSR[MAV, LD, BUS_DRERR].
- The error is caused by writing to the area affected by the ECC multibit error sets MCSR[MAV, LD, BUS_DRERR, BUS_WRERR]. This can be achieved by 32-bit, 16-bit, and 8-bit writes, because it behaves as a read-modify-write operation above 64bits.
- The error is caused by an attempt to execute an instruction affected by the ECC multibit error sets MCSR[MAV, IF, BUS_IRERR].
- The error is caused by the cache-line-filling set MCSR[MAV, BUS_DRERR] or MCSR[MAV, BUS_IRERR] when there are data affected by the ECC multibit error within this line, but not in the currently loaded data.

MCSR[MAV] - indicates that the address in the MCAR was updated by hardware. Note that the next update is only performed when this bit is explicitly cleared by the W1C operation (Write 1 to Clear).

8.1.2. Machine Check Address Register (MCAR)

This register contains the effective (when MCSR[MEA]=1) or physical (when MCSR[MEA]=0) addresses, for which the asynchronous type of the machine check exception was raised (not all machine sources update this register).

The address is valid only when MCSR[MAV] was cleared before the exception.

8.1.3. Machine Check Save/Restore Register 0 (MCSRR0)

This register contains the address of the instruction that caused the exception. At the end of the exception service routine, the “rfmci” instruction loads the content of this register as the return address (program counter).

To return to the program flow before the exception occurred (i.e. after the instruction that caused the exception), increase the exception returning address by the length of instruction causing an exception (in case of BookE by 4, in case of VLE by 2 or 4, according to the instruction opcode).

For details, see the *VLE 16-bit and 32-bit Instruction Length Decode Algorithm* (document AN4648).

8.2. Interrupt handling of non-correctable ECC error

According to the implementation, the MPC57xx devices have either the MEMU or ERM modules.

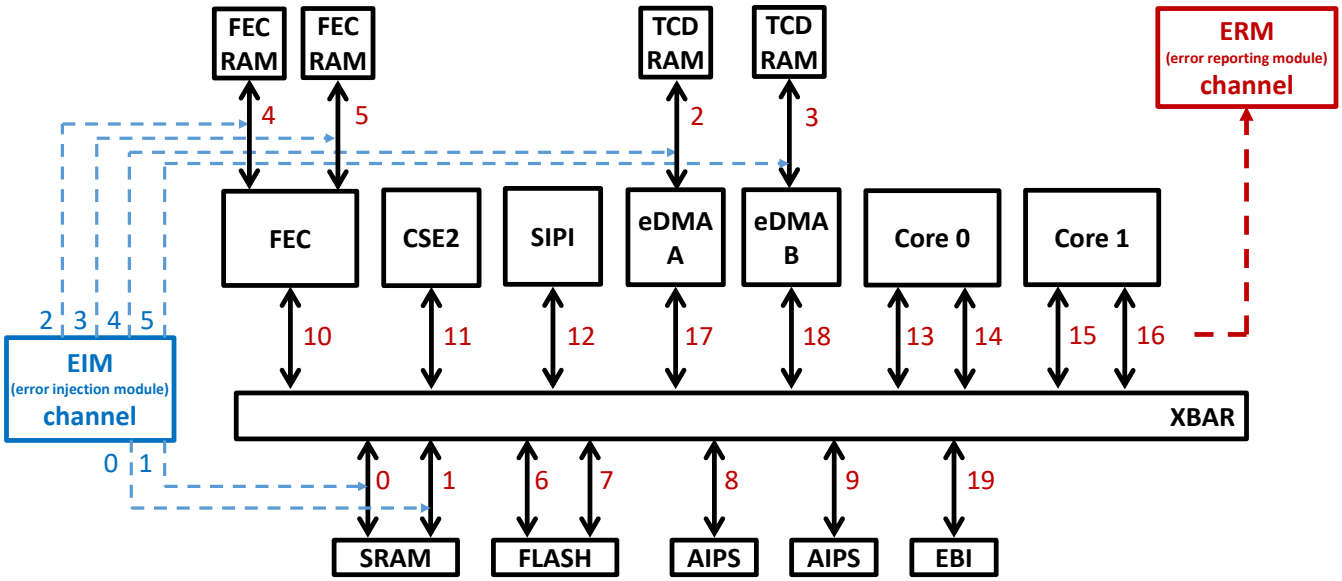
Table 3. Used reporting module

	Memory Error Management Unit (MEMU)	Error Reporting Module (ERM)
MPC5744P	Yes	No
MPC5746C	Yes	No
MPC5748G	Yes	No
MPC5746R	Yes	No
MPC5775K	Yes	No
MPC5777C	No	Yes
MPC5775B/E	No	Yes
MPC5777M	Yes	No

The following two sections briefly describe the MPC5777C’s and MPC5775B/E’s ERM and FCCU modules, because it is necessary to understand their functionality to create an application-specific ECC/EDC handler.

8.2.1. ERM

Error reporting module



8.2.2. **FCCU**

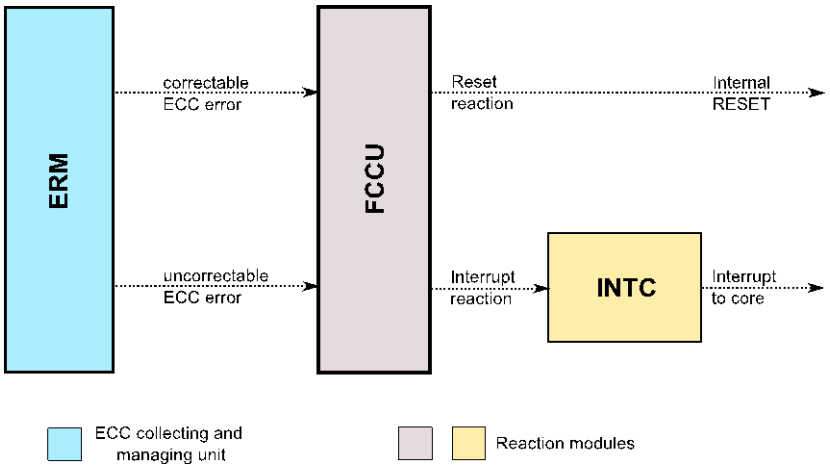


Figure 1. MPC5777C's ECC error reporting path

9. ECC error injection methods

9.1. Intentional generation of SRAM 1b/2b ECC error

Table 4. Options for 2b embedded RAM ECC error injection across MPC57xx product line

	Error injection module (EIM)	SRAM ECC Error injection with E2EECSR (DCR 511)	Indirect Memory Access (IMA)
MPC5744P	for DMA TCD RAM array	Yes	No
MPC5746C	No	Yes	No
MPC5748G	No	Yes	No
MPC5746R	for DMA TCD RAM array	Yes	No
MPC5775K	for DMA TCD RAM array	Yes	No
MPC5777M	No	Yes	Yes
MPC5777C MPC5775B/E	for DMA TCD RAM array, FEC RAM array, internal SRAM array	No	No

9.1.1. EIM

TBD

9.2. Intentional generation of FLASH 2b ECC error

9.2.1. Option 1 - flash over programming

ECC errors can be generated in the flash by over-programming memory locations.

A multiple bit error is detected but not corrected. Therefore, it is easier to see when it is injected. A procedure for creating a multiple bit error is as follows:

1. Write the original data A = 0x0045000000000000 to a flash memory location.
2. Over-program data A to data B = 0x0058000000000000 in the same flash memory location.

This creates a multiple-bit ECC error and it is flagged in the ECC module.

Make sure to choose data patterns that have different ECC checkbits, because over-programming does not necessarily generate an ECC error. For instance, the patterns shown in Table 9 have the same ECC checkbits and they can be overwritten without generating an ECC error (this feature is utilized by EEPROM emulation drivers).

Table 5. Data patterns with same ECC checkbits

Doubleword	Checkbits
0xFFFF_FFFF_FFFF_FFFF	0xFF
0xFFFF_FFFF_FFFF_0000	0xFF
0xFFFF_FFFF_0000_FFFF	0xFF
0xFFFF_0000_FFFF_FFFF	0xFF
0x0000_FFFF_FFFF_FFFF	0xFF
0xFFFF_FFFF_0000_0000	0xFF
0xFFFF_0000_FFFF_0000	0xFF
0x0000_FFFF_FFFF_0000	0xFF
0xFFFF_0000_0000_FFFF	0xFF
0x0000_FFFF_0000_FFFF	0xFF
0x0000_0000_FFFF_FFFF	0xFF
0xFFFF_0000_0000_0000	0xFF
0x0000_FFFF_0000_0000	0xFF
0x0000_0000_0000_0000	0xFF

9.2.2. Option 2 - reading of UTEST area

The MPC57xx device family offers a set of test patterns, pre-programmed in the factory (see Table 6). For a multi-bit error, read address 0x00400060.

Table 6. UTEST flash memory (extract)

Start address	End address	Size (bytes)	Description	Notes
0x00400040	0x0040005F	32	Customer Single Bit Correction Area	*
0x00400060	0x0040007F	32	Customer Double Bit Detection Area	*
0x00400080	0x0040009F	32	Customer EDC after ECC Area	*

* Programmed by NXP to include ECC/EDC errors to allow testing of ECC/EDC hardware.

9.3. Intentional generation of FLASH 1b ECC error

The 1b ECC error is corrected without reporting it into the MEMU/ERM module by default (C55FMC_MCR[SBC] is not set on the 1b ECC event). Enable it by configuring the PFLASH controller using the following sequence:

```
/* Enable single bit ECC error reporting in flash controller */
// Enable UTest mode
C55FMC.UT0.R = 0xF9F99999;
// Enable single bit error correction
C55FMC.UT0.B.SBCE = 1;
// Finish the UTest mode by writing UT0[UTE] with 0.
C55FMC.UT0.B.UTE = 0;
```

9.3.1. Option 1 - flash over programming

The approach described in Option 1 - flash over programming can be used. The chosen pattern for a single-bit injection may be as follows:

1. Write the original data A = 0xFFFFFFFF00000000 (syndrome X) to a flash memory location.

2. Over program data A to data B = 0xFFFFFFFF00000001 (syndrome Y) to the same flash memory location.

9.3.2. Option 2 - reading of UTEST area

The MPC57xx device family offers a set of test patterns pre-programmed in the factory (see Table 6). For a single-bit error, read address 0x00400040.

9.4. Intentional generation of Flash EDC after ECC error

10. Example code

10.1. MPC5777C 1b+2b FLASH ECC error injection

Purpose of the example is to show how to generate Multi-bit or Single-bit ECC error in internal FLASH (user must choose it in the option at the end of main function).

Flash over-programming is used to generate a non-correctable (or single-bit) ECC error in FLASH. The bad data is accessed then, so the IVOR1 exception (or ERM combined interrupt service routine) is generated and handled.

Example also offers useful macros for EIM and ERM modules.

The example displays notices in the terminal window (USBtoUART bridge J21)

(19200-8-no parity-1 stop bit-no flow control on eSCI_A).

No other external connection is required.

10.2. MPC5777C 1b+2b RAM ECC error injection

Purpose of the example is to show how to generate Multi-bit or Single-bit ECC error in internal SRAM (user must choose it in the option at the end of main function).

Error Injection Module is used to generate a non-correctable (or single-bit) ECC error in RAM. The bad data is accessed then, so the IVOR1 exception (or ERM combined interrupt service routine) is generated and handled.

Example also offers useful macros for EIM and ERM modules.

The example displays notices in the terminal window (USBtoUART bridge J21)

(19200-8-no parity-1 stop bit-no flow control on eSCI_A).

No other external connection is required.

RoHS

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.

Document Number: ANxx
Rev. 0 Draft A
09/2021

arm

NXP