Freescale Semiconductor
Application Note

Document Number: AN5200
Rev. 1, 12/2015

Error Correcting Codes Implemented on MPC55xx and MPC56xx Devices

by: David Tosenovjan

Error Correction Code, or 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.

All of the ECC methods used with MPC55xx and MPC56xx devices provide Single Error Correction - Double Error Detection (SECDED) capability.

Intention of this application note is to describe how ECC protection is implemented with MPC55xx and MPC56xx devices and understand particular MCU's ECC event response. It also offers some software examples for illustration of mentioned behavior.

Table of Contents

1 ECC protected memory's initialization

1.1 SRAM initialization after Power-on-reset

Reset state of internal SRAM is random, thus data and checkbits may contain any data. Most probably the first read attempt to any address would generate non-correctable ECC error. Thus SRAM must be initialized after power-up. Typically it means the whole SRAM is deleted or written by any value, however it must be either 64-bit write or 32-bit write (according to Table 2), to completely define ECC code for data unit. Otherwise smaller sized write access results to read-modify-write with another ECC error.

```
/* MPC5674F L2SRAM initialization code */
lis r11,L2SRAM_LOCATION@h /* Base address of the L2SRAM, 64-bit word aligned */
ori r11,r11,L2SRAM_LOCATION@l

li r12,2048 /* Loop counter to get all of L2SRAM */
mtctr r12

init_l2sram_loop:
    stmw r0,0(r11)        /* Write all 32 GPRs to L2SRAM */
    addi r11,r11,128      /* Inc the ram ptr; 32 GPRs * 4 bytes = 128 */
    bdnz init_l2sram_loop /* Loop for 256k of L2SRAM */
```

1.2 SRAM initialization after Synchronous reset

User can omit SRAM initialization for synchronous reset sources - on most devices it is only software reset (see particular device's Reference Manual for details). For asynchronous reset there is certain risk of inducing multibit ECC error if device has been reset during SRAM write.

Thus for asynchronous reset sources application should initialize SRAM, even though content of SRAM is kept untouched over reset.

1.3 Initialization of other embedded SRAM memories

Most MPC55xx/56xx devices have ECC protection implemented only for internal FLASH and SRAM. However trend started with MPC5676R device is to have ECC algorithm implemented on all SRAM based internal memories, including Shared Code Memory (SCM) and Shared Parameter RAM (SPRAM) of eTPU modules and FlexRay message buffers. Note that access to this content may lead in ECC event, if it is not previously initialized.

2 Used ECC Algorithms

2.1 Internal FLASHes

Employed ECC algorithms (see Table 1) are ‘modified’. The modification lies in algorithm change the way that ECC checkbits 0xFF belongs to data 0xFFFF_FFFF_FFFF_FFFF. Thus fully erased flash memory is not in corrupted state.

Table 1. ECC Algorithms on internal FLASH memories

	Modified Hamming Code (64+8)	Hsiao Code (64+8)	Hsiao Code (32+7)
Code or unified flash	• MPC55xx • MPC5668G/E • MPC567xF • MPC5644A • MPC5602B/3B/4B • MPC5602C/3C/4C • MPC5603P/4P • MPC560xS • MPC563xM • MPC5605B/6B/7B (only Rev.1) • PXD10 • PXN20 • PXR40	• MPC564xS • MPC564xL • MPC5642A • MPC5676R • MPC5605B/6B/7B (Rev.2 and higher) • MPC564xC • MPC5601D/02D • MPC5601P/2P • MPC567xK • MPC560xE • PXS20 • PXS30 • PXD20	• none
Data flash	• MPC5602B/3B/4B • MPC5602C/3C/4C • MPC5603P/4P • MPC560xS, • MPC563xM, • MPC5605B/6B/7B (only Rev.1) • PXD10	• MPC5605B/6B/7B (Rev.2 and higher)	• MPC5601D/02D • MPC5601P/2P • MPC564xC • MPC567xK • MPC560xE • PXS30

2.2 Internal SRAMs

Internal SRAMs use Modified Hamming Code or Hsiao Code in configuration (64+8) or (32+7).

Table 2. ECC Algorithms on internal SRAM memories

	(64+8)	(32+7)
Internal SRAM	• MPC5553 • MPC5554 • MPC5561 • MPC5565 • MPC5566 • MPC5567 • MPC5674F • MPC5676R • MPC5675K • PXS30 • PXR40	• MPC553x • MPC551x • MPC5604B/C • MPC5602D • MPC5607B • MPC5604P • MPC5602P • MPC5606S • MPC5645S • MPC5604E • MPC5668GE • MPC5634M • MPC5644A • MPC5642A • MPC5643L • MPC5646B/C • PXD10 • PXN20 • PXD20 • PXS20

Certain devices have SRAM organized as 2x(32+7) according to Table 3. It must be taken into account during ECC error injection (see section 6.1)

Table 3. MPC5643L, MPC5644A and MPC5646B/C SRAM organization

Bits RAM	EVEN [31-24]	EVEN [23-16]	EVEN [15-8]	EVEN [7-0]	ODD [31-24]	ODD [23-16]	ODD [15-8]	ODD [7-0]
ERRBIT	[63-56]	[55-48]	[47-40]	[39-32]	[31-24]	[23-16]	[15-8]	[7-0]
Doubleword Address	0x0 - 0x3 0x8 - 0xB ...				0x4 - 0x7 0xC - 0xF …			

Bits Parity	EVEN [6-0]	ODD [6-0]
ERRBIT	[76-71]	[70-64]
Doubleword Address	0x0 - 0x3 0x8 - 0xB ...	0x4 - 0x7 0xC - 0xF …

3 Behavior in case ECC event occurs

3.1 z0, z1, z3, and z6 versions of e200 cores

Single bit errors are automatically corrected - no reporting.

Detection of Multiple Bit ECC errors causes one of following:

Table 4. Multiple Bit ECC Error detection for z0/z1/z3/z6

MSR[EE]	MSR[ME]	Access Type	Result
0	0	Instruction or data	Enter Checkstop state
0	1	Instruction or data	Machine Check Interrupt (IVOR 1)
1	x	Data	Data Storage Interrupt (IVOR2) * External Termination Error bit is set in spr ESR
1	x	Instruction	Instruction Storage Interrupt (IVOR3) *

and (if enabled)

External Interrupt (IVOR4) – ECSM/MCM combined interrupt request (usually vector 9)

- in case ECC error is invoked by other master then core, for instance eDMA, only ECSM interrupt will be initiated
- in case ECC error is invoked by the core, both ECSM interrupt and particular exception will be initiated, in sequence according to their priorities.

* MPC5553/5554 - erratum e1143 - Option with cleared MSR[EE] bit is not functional and ECC non-correctable error will act as MSR[EE] would be set.

3.2 z4 and z7 versions of e200 cores

Single bit errors are automatically corrected but can be reported to ECSM module.

Detection of Multiple Bit ECC Errors causes one of following:

Table 5. Multiple Bit ECC Error detection for z4/z7

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

and (if enabled)

External Interrupt (IVOR4) – ECSM/MCM combined interrupt request (usually vector 9)

- in case ECC error is invoked by other master then core, for instance eDMA, only ECSM interrupt will be initiated
- in case ECC error is invoked by the core, both ECSM interrupt and particular exception will be initiated, in sequence according to their priorities.

* Checkstop state does not exist on z4 and z7 cores

3.3 FLASH Non-correctable error reporting

On certain devices, PFBIU behave the way it invalidates the entire page (128 bits) when there's a non-correctable ECC error in any of the double-word in that page - these are all MPC56xx devices with C90LC flash + MPC5668G/E (or PXN20). Other products (with C90FL flash) invalidates only faulty double word when non-correctable ECC error occur. In other words, read operation is always 128 bits on devices highlighted in Table 6.

Table 6. Internal flash types

Flash type	H7Fa	H7Fb	C90LC	C90FL
Device family	• MPC5553 • MPC5554 • MPC5561 • MPC5565 • MPC5566 • MPC5567	• MPC5533 • MPC5534 • MPC551x	• MPC560xB • MPC560xP • MPC560xS • MPC560xE • MPC563xM • MPC564xB • MPC564xC • MPC567xK • PXD10 • PXS30	• MPC564xA • MPC564xL • MPC564xS • MPC5668x • MPC567xF • MPC5676R • PXN20 • PXD20 • PXS20 • PXR40

3.4 Impact of enabled CACHE on error reporting

In case cache is enabled, there is attempt to fill the whole cache line (32 bytes) as it is burst operation. If there is a non-correctable ECC error within this line, the line is invalidated and ECC related exception is invoked. For details, see section 5.1.1.

4 Correctable ECC error servicing

Basically it is not needed as 1-bit ECC error are automatically corrected during data/instruction read.

Detection of 1-bit errors (only MPC56xx) could be useful for catching of gradual degradation of flash memory content caused by aging. 1-bit error may be “physically” repaired by reading the data and writing them back.

5 Non-correctable ECC error servicing

Fixing of multibit ECC error is application dependent and must be part of ECC error interrupt handling. It can be based IVOR1/2/3 exception handler or IVOR4 interrupt handler as described in section 5.1 and section 5.2.

5.1 Based on IVORx exception

This document describes IVOR1 (Machine check) handling as the most common one and available with all e200 cores. The precondition is to have MSR[ME]=1 as it guarantees exception will be taken not

only directly associated with current instruction execution stream, but also those reported by the subsystem as bus error termination (for example cache line filling). This approach however still catch only error related to the core.

If ECC non-correctable error caused by other master than the core is supposed to be serviced, handling based on ECSM interrupt must be used (section 5.2) or combination of both approaches.

5.1.1 Machine Check Syndrome Register (MCSR)

Register gives possibility to differentiate between sources of machine check exceptions. Exception service routine should analyze the root cause of exception:

- Error caused by read of ECC corrupted data sets MCSR[MAV, LD, BUS_DRERR].
- Error caused by write to area affected by ECC multibit error (this can be achieved by 32-bit, 16-bit and 8-bit write as it behaves as read-modify-write operation above 64-bit) sets MCSR[MAV, LD, BUS_DRERR, BUS_WRERR]
- Error caused by attempt to execute of instruction affected by ECC multibit error sets MCSR[MAV, IF, BUS_IRERR].
- Error caused by cache line filling (when there are data affected by ECC multibit error within this line) set MCSR[MAV, BUS_DRERR] or MCSR[MAV, BUS_IRERR].

MCSR[MAV] - indicates that the address contained in the MCAR was updated by hardware. Note that next update is only performed when this bit is explicitly cleared by w1c operation („write 1 to clear“).

5.1.2 Machine Check Address Register (MCAR)

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

Address is valid only when MCSR[MAV] was cleared before exception.

5.1.3 Machine Check Save/Restore Register 0 (MCSRR0)

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

In case it is needed to return to program flow before exception (i.e. after instruction that caused the exception) then it is needed to increase exception returning address by length of instruction causing an exception (in case BookE by 4, in case VLE by 2 or 4 according to instruction opcode) – see application note AN4648.

Table 7. Read content of address given by MCSRR0 register during machine check exception

Bit3	Bit0	Instruction was	Increment MCSRR0 by
0	0	16-bit	2
0	1	16-bit	2
1	0	32-bit	4
1	1	16-bit	2

NOTE

Address of instruction causing an IVOR1 exception (machine check) is stored is MCSRR0 register only with devices with “new” cores (z4 and z7). MCSRR0/1 registers are not present with “old” cores (z0, z1, z3, z6) and this information is stored in SRR0 register. Also se_rfci instruction must be used instead of se_rfmci.

5.2 Based on ECSM interrupt

5.2.1 ECSM module

A part of Error Correction Status Module (ECSM) is dedicated to ECC handling. It provides set of registers offering extended information about last detected ECC event. In comparison to approach described in section 5.1, ESCM brings following benefits:

- is capable to distinguish between RAM and FLASH ECC event
- allows detection of ECC event from other XBAR masters then the Core (eDMA, ..)
- can detect also single bit correctable ECC events (only MPC56xx)
- offers RAM ECC error injection capability

Table 8. ECSM module's ECC related registers

Regíster Description	Register
ECC Configuration Register	ECSM_ECR
ECC Status Register	ECSM_ESR
ECC Error Generation Register	ECSM_EEGR
Flash ECC Registers	ECSM_FEAR ECSM_FEMR ECSM_FEAT ECSM_FEDR
RAM ECC Registers	ECSM_REAR ECSM_RESR ECSM_REMR ECSM_REAT ECSM_REDR

5.2.2 Coherent software view of the reported ECC event

Following sequence clears particular flags in ECSM_ESR register by recommended sequence to maintain the coherent software view of the reported ECC event:

```
for (;;)
{
    /* 1. Read the ECSM_ESR and save it */
    ecsm_esr = (uint8_t)ECSM.ESR.R;

    /* 2. Read and save all the address and attribute reporting registers */

    /* Read only RAM registers if error is caused by RAM ECC error */
    if( (ecsm_esr & ECSM_ESR_R1BC_MASK) || (ecsm_esr & ECSM_ESR_RNCE_MASK) )
    {
        ecsm_rear = (uint32_t)ECSM.REAR.R;
        ecsm_remr = (uint8_t)ECSM.REMR.R;
        ecsm_reat = (uint8_t)ECSM.REAT.R;
    }

    /* Read only FLASH registers if error is caused by FLASH ECC error */
    if( (ecsm_esr & ECSM_ESR_F1BC_MASK) || (ecsm_esr & ECSM_ESR_FNCE_MASK) )
    {
        ecsm_fear = (uint32_t)ECSM.FEAR.R;
        ecsm_femr = (uint8_t)ECSM.FEMR.R;
        ecsm_feat = (uint8_t)ECSM.FEAT.R;
    }

    /* 3. Re-read the ECSM_ESR and verify the current contents matches
```

```
           the original contents. If the two values are different, go back
           to step 1 and repeat */
    if (ecsm_esr == (uint8_t)ECSM.ESR.R)
    {
        break;
    }
}

/* 4. When the values are identical, write a 1 to the asserted ECSM_ESR flag
      to negate the interrupt request - we actually negate all flags */
ECSM.ESR.R = 0x33;
```

6 ECC error injection

6.1 Intentional generating of SRAM 2b ECC error

SRAM double-bit ECC error may be injected by ECSM module. The whole sequence (injection and handling) is being done by following steps:

1) To enable ECC reporting:

```
/* enable RAM ECC error reporting */
ECSM.ECR.R = (uint8_t)ECSM_ECR_ERNCR_MASK | // non-correctable RAM ECC error
                      ECSM_ECR_ER1BR_MASK ; // 1-bit RAM ECC error
```

2) To generate non-correctable ECC error:
   a) enable generating of ECC error
   b) write data to particular RAM location
   c) read the particular location to assert ECC exception

```
/* align this to a 0-modulo-8 address (aligned to 8 bytes) */
static vuint32_t test[2] __attribute__ ((aligned (8))) =
{ 0x0ul, 0x0ul };

static vuint32_t test_read = 0;

ECSM.EEGR.R = (uint16_t)(ECSM_EEGR_FR1NCI_MASK | 0x0001); // RAM[1] is inverted
test[0] = 0xCAFEBEEF;
test_read = test[0];
```

3) To service ECC exception according (Section 5.1) and/or (Section 5.2)

6.2 Intentional generating of FLASH 2b ECC error

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

A multiple bit error will be detected but not corrected, therefore it is easier to see it has occurred. 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 to the same flash memory location.

This will create a multiple bit ECC error and will be flagged in the ECC module.

Note that it is needed to choose such data patterns that have different ECC checkbits because over-programming does not necessarily generates ECC error. For instance, patterns shown in Table 9 has the same ECC checkbits and thus may be overwritten without generating of ECC error. Moreover this feature is utilized by EEPROM emulation drivers.

Table 9. 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

7 ECSM/RGM/FCCU relation on safety devices

This section is related only to MPC564xL and MPC567xK devices and describes their specific implementation. ECC error is now also reported to FCCU and RGM modules as shown in the Figure 1. ECSM detection of multibit ECC error also cause FCCU critical faults CF16 and CF17.

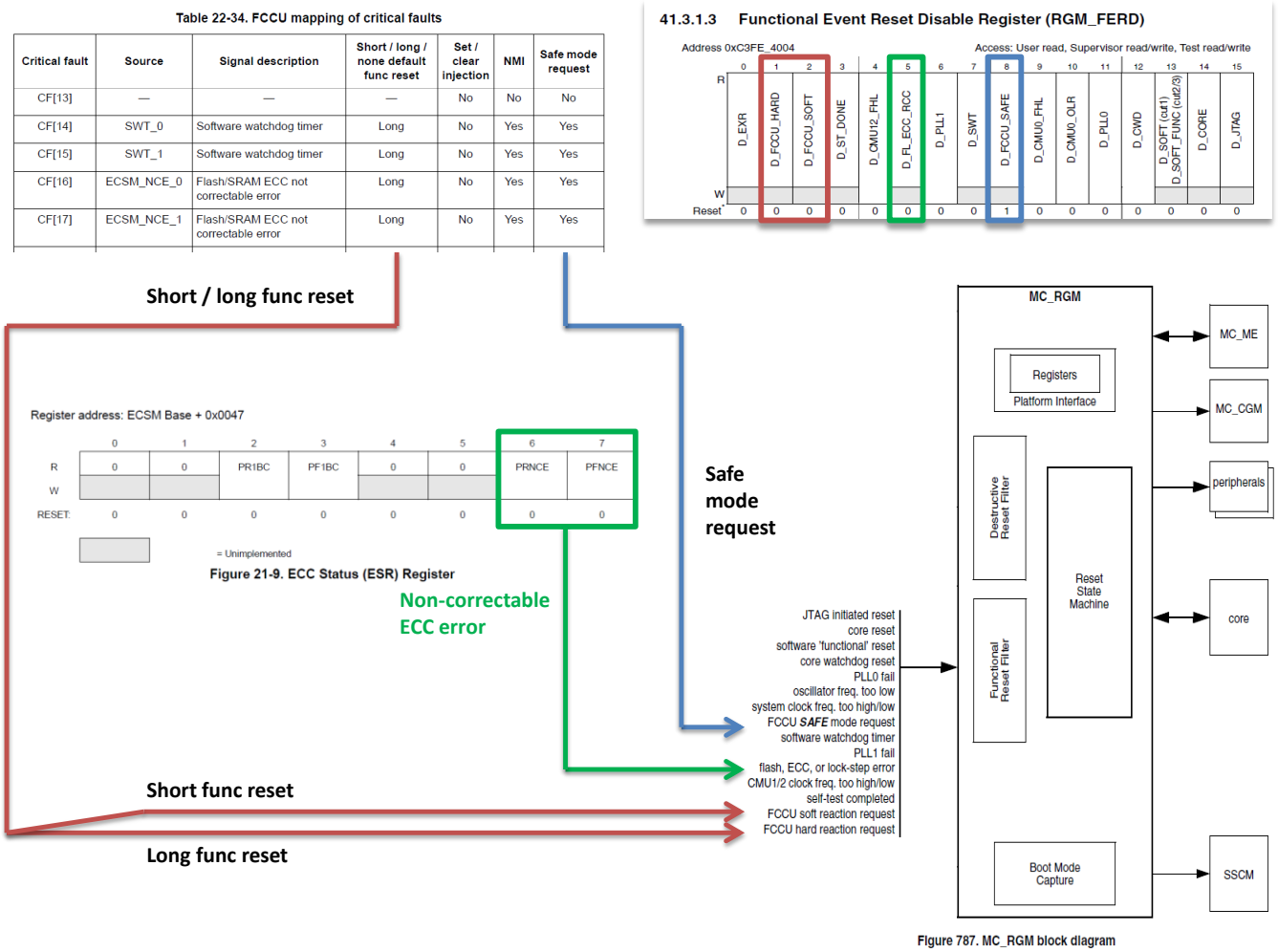

Figure 1. Reset Generation Module reporting on MPC564xL

7.1 MPC564xL

- Non-correctable ECC error reporting is directly routed from ECSM into RGM module via RGM_FES[F_FL_ECC_RCC] flag. RGM_FERD[D_FL_ECC_RCC] and RGM_FESS[SS_FL_ECC_RCC] bits are read-only and thus not configurable what means that flash, ECC, or lock-step error event always triggers a reset sequence starting from PHASE1. This cannot be changed on MPC564xL device.
- If ECSM_ ECR[EPRNCR/EPFNCR] bit is set and so ECSM Non-Correctable Reporting is enabled and multibit ECC error even occurs (either unintentional or injected), device is reset.

- Platform Flash ECC registers (PFEAR, PFEMR, PFEAT, PFEDRH, PFEDRL) and Platform RAM ECC registers (PREAR, PRESR, PREMR, PREAT, PREDRH, PREDRL) are maintained through the MCU functional reset. This can be however used only with ECC error injection - because ECSM_ECR is being cleared during reset, user cannot distinguish whether the reason was ECC error in FLASH or RAM. Although these registers are maintained, only the half of them is readable. For instance if Multibit ECC error occurs in RAM, device goes to reset and then Platform RAM ECC registers are maintained. But Platform FLASH ECC registers contains random data (and different for ECSM_0 and ECSM_1) so attempts to read any of Platform FLASH ECC register (in order to distinguish between RAM and FLASH error) leads in lock-step error and another reset. And vice versa when Multibit ECC error occurs in RAM, Platform FLASH ECC registers are not usable.

7.1.1 Handling through reset

```
/* Save RGM status register before initialization for subsequent usage */
RGM_FES_reg = RGM.FES.R;

/* Initialize HW common way */
HW_init();

/* check ECC error */
if(RGM_FES_reg & RGM_FES_F_FL_ECC_RCC)
{
     /* apply ECC error handling */
}
```

7.1.2 Possible ECC error handling through reset solution

7.1.2.1 Option 1

- After reset caused by ECC error, do not enable ECSM reporting
- Scan the whole flash memory range (0-0x00F0_3FFF) and handle possible ECC error in Machine check exception (IVOR1)
- If none error occurs, consider there was an error in RAM, already corrected by SRAM initialization
- Pros and cons:

  \+ catches all multibit ECC errors from all masters

  – slow recover due to reset and impossibility to simply find an error

7.1.2.2 Option 2

- Do not use ESCM reporting at all
- Pros and cons:
  + possible fast recover by IVOR1 exception
  – does not catch ECC errors from other masters than the core
  – it is not possible to inject Multibit RAM ECC error

7.2 MPC567xK

The same approach as described in section 7.1 can be used as well.

Contrary to MPC564xL, RGM_FERD[D_FL_ECC_RCC] bit is writable and MPC567xK device can be configured the way not to generate reset in case double bit ECC error occurs.

If RGM_FERD[D_FL_ECC_RCC] = 1, ECC double bit error generates safe mode request and device goes to machine check exception. However this machine check is invoked two times, firstly by Read Bus Error, secondly by FCCU non-maskable interrupt (delay between this two events is not precisely defined as Core and FCCU module are driven by different and independent clock sources). Multiply IVOR1 issuing cause loosing of returning address (MCSRR0) thus this exception is non-recoverable and application must be reset anyway by software reset.

8 Example Codes

8.1 MPC5634M 2b_RAM_ECC_error_injection

ECSM Error Generation Register is used to generate a non-correctable ECC error in RAM. The bad data is accessed then, so the IVOR1/2/3 exception is generated and handled. It shows also ECSM combined interrupt service routine and how to correct the wrong data.

Use macro Induce_ECC_error_by_DMA_read to select whether ECC error will be injected by DMA read or CPU read.

At the end of main file you can select particular ME/EE setup by comment/uncomment of particular function calls.

8.2 MPC5674F 1b+2b_RAM_ECC_error_injection

ECSM Error Generation Register 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 1b ECC error handler) is generated and handled. It shows also ECSM combined interrupt service routine and how to correct the wrong data.

Use macro Induce_ECC_error_by_DMA_read to select whether ECC error will be injected by DMA read or CPU read.

At the end of main file you can choose if single bit or multi bit is injected and select particular ME/EE setup by comment/uncomment of particular function calls.

8.3 MPC5643L 1b_RAM_ECC_error_injection

Example demonstrates MCU behavior when single bit RAM ECC error occurs by intentional ECC error injection.

8.4 MPC5643L 2b_RAM+2b_FLASH_ECC_error_injection

Purpose of the example is to show how to generate Multi bit ECC error in internal SRAM or FLASH (user must choose it in the option at the end of main function) and how to handle this error with respect to constraints given by MPC5643L architecture (ECSM/RGM/FCCU relation and ECC error handling through reset). The example is only possible to run in internal_FLASH target. Power-on-reset is required after downloading the code into MCU's flash. The example displays notices in the terminal window (19200-8-no parity-1 stop bit-no flow control on eSCI_A). No other external connection is required.

8.5 MPC5675K 2b_RAM+2b_FLASH_ECC_error_injection

Purpose of the example is to show how to generate Multi bit ECC error in internal SRAM or FLASH (user can choose it in the option at the end of main function) and how to handle this error with respect to constraints given by MPC5675K architecture (ECSM/RGM/FCCU relation and ECC error handling through reset). The example is only possible to run in internal_FLASH target. Power-on-reset is required after downloading the code into MCU's flash. The example displays notices in the terminal window (19200-8-no parity-1 stop bit-no flow control on eSCI_A). No other external connection is required.

Example also shows impact of enabled cache (macro OPTIMIZATIONS_ON).

How to Reach Us:

Home Page:
freescale.com

Web Support:
freescale.com/support

Document Number: AN5200
Rev.1
12/2015