AN14960

CM4 TCM ECC Validation on i.MX RT1170

Rev. 2.0 — 11 September 2026 Application note

Document information

Information	Content
Keywords	AN14960, i.MX RT1170, CM4, ECC
Abstract	This document details a method to validate CM4 TCM ECC on i.MX RT1170.

1 Introduction

For i.MX RT1170 devices, NXP provides a general introduction to Error-Correcting Code (ECC) usage in *i.MX RT1170 ECC Application* (document AN13204). However, the CM4 TCM ECC block is special as it does not support ECC error injection directly for validation.

This document details a method to validate CM4 Tightly-Coupled Memory (TCM) ECC on i.MX RT1170.

2 How to design the CM4 TCM ECC validation

As the CM4 TCM ECC does not support direct error injection, validating CM4 TCM ECC requires an understanding of how the CM4 TCM ECC IP logic operates.

2.1 ECC operation principles

To validate CM4 TCM ECC behavior, the following basic ECC principles must be understood:

- The TCM word level ECC uses a Hsiao odd-weight column criteria ECC code for ECC generation and checking. It uses a 7-bit ECC code word per word (32-bits) of data. The final ECC code is XORed with 0x38 before writing into ECC code memory.
- Fuse 0x840[2] must be burned to enable CM4 TCM ECC.
- ECC enabled: When ECC is enabled and a 32-bit data is written to TCM, an additional 7-bit ECC code is generated. The 32-bit data and its 7-bit ECC code are then written to the target memory.
- ECC disabled: When ECC is disabled and a 32-bit data is written to TCM, the corresponding 7-bit ECC code memory region is simultaneously filled with 000_0000b at the same time.

2.2 Validation requirements

Based on the principles mentioned in Section 2.1, the following steps are required to validate CM4 TCM ECC:

1. Identify a 32-bit data value V1 whose ECC code is 0x38. One of the possible values of V1 is 0x8000044C.
2. Generate a 1-bit correctable ECC event:
   a. Disable ECC.
   b. Flip 1 bit in V1 and write the modified data to the target address.
   c. Then re-enable ECC and read the target address.
   d. The operation above must trigger a 1-bit correctable ECC event.
3. Generate a 2-bit uncorrectable ECC event:
   a. Disable ECC.
   b. Flip 2 bits in V1 and write the modified data to the target address.
   c. Then re-enable ECC and read the target address.
   d. The operation above must trigger a 2-bit uncorrectable ECC event.

2.3 Interrupt handling behavior

In the SDK implementation:

- A 1-bit ECC event triggers `CORE_IRQHandler()`.
- A noncorrectable event triggers `NMI_Handler()`.

If Parity Fault Enable (PFE) in MCM_LMDRn is also enabled, when noncorrectable event occurs, `HardFault_Handler()` is also triggered. When a noncorrectable event occurs, the ISR(either NMI or

Hardfault) must fix the error in the address indicated by LMFAR. Otherwise, when the ISR returns, it reexecutes the previous instruction, which triggers a noncorrectable event, leading to an infinite loop.

3 Code design

The code performs three validation tests:

- When ECC is enabled, a value is read from an ECC enabled and preloaded memory region. There must be no 1-bit correctable event or uncorrectable event occur.
- Test a 1-bit correctable ECC event using the method mentioned in Section 2.
- Test a 2-bit uncorrectable ECC event using the method mentioned in Section 2.

Note: *For code details, refer Section 5. Place this code into a CM4 Hello World example from the i.MX RT1170 SDK, avoid using the CM4 TCM in the linker file for testing purposes, and then build and run.*

4 Log

The terminal output for each test is provided below.

Test 1 log: The following log shows the result of test 1, where a clean memory read is performed.

```
CM4 ECC validate test.
UUID:
40cac900: 57ac5969
40cac910: b86db00b
Test 1:
Done.
PASS.
```

Test 2 log: The following log shows the result of test 2, where a 1-bit correctable ECC event is generated.

```
Test 2:
Read addr: 20000000
CORE_IRQHandler
MCM->LMFAR = 20000000
MCM->LMFATR = 430023
MCM->LMFDHR = 0
MCM->LMFDLR = 8000044d
correctable inerrupt happen.
MCM->LMPEIR: 89000200
a 1 bit -correctable ECC event from system TCM
1 bit Error detected on system TCM
Exit CORE_IRQHandler.
Done.
v = 8000044c
PASS.
```

Test 3 log: The following log shows the result of test 3, where a 2-bit uncorrectable ECC event is generated.

```
Test 3:
Read addr: 20000000
----------
NMI_Handler
MCM->LMFAR = 20000000
MCM->LMFATR = 60023
MCM->LMFDHR = 0
```

```
MCM->LMFDLR = 8000044f
uncorrectable inerrupt happen.
MCM->LMPEIR: 81000002
a non-correctable ECC event from system TCM
Non-correctable Error detected on system TCM
exit uncorrectable_callback
Exit NMI_Handler
Done.
v = 8000044f
PASS.
```

5 Appendix

The code below implements ECC enable/disable control, event handling, and the three validation tests described in this document.

```
#include "fsl_device_registers.h"
#include "fsl_debug_console.h"
#include "board.h"
#include "app.h"

/*******************************************************************************
 * Definitions
 ******************************************************************************/

/*******************************************************************************
 * Prototypes
 ******************************************************************************/

/*******************************************************************************
 * Variables
 ******************************************************************************/

/*******************************************************************************
 * Code
 ******************************************************************************/
/*!
 * @brief Main function
 */
void clear_status(void)
{
    MCM->LMPEIR = 0xffffff;

    asm(" dsb");
}
void show_reg(void)
{
    PRINTF("MCM->LMDR[0]: %x \r\n", MCM->LMDR[0]);
    PRINTF("MCM->LMDR[1]: %x \r\n", MCM->LMDR[1]);
}
void enable_tcm_ecc(void)
{
    // Enable ECC for read, write for code TCM
    MCM->LMDR[0] |= MCM_LMDR_CF0(3);

    // Enable ECC for read, write for system TCM
    MCM->LMDR[1] |= MCM_LMDR_CF0(3);
```

```
    // Enable 1 bit/uncorrectable ECC interrupt and reporting.
    MCM->LMPECR |= MCM_LMPECR_ERNCR_MASK | MCM_LMPECR_ERNCI_MASK |
 MCM_LMPECR_ER1BR_MASK | MCM_LMPECR_ER1BI_MASK;
}
void disable_tcm_ecc(void)
{
    MCM->LMDR[0] &= ~MCM_LMDR_CF0(3);
    MCM->LMDR[1] &= ~MCM_LMDR_CF0(3);
}
void decode_status(void)
{
    int v = MCM->LMPEIR;

    PRINTF("MCM->LMPEIR: %x \r\n", MCM->LMPEIR);

    int peeloc = (MCM->LMPEIR>>24)&0x1f;

    if(peeloc == 0)
    {
        PRINTF("a non-correctable ECC event from code TCM \r\n");
    }
    if(peeloc == 1)
    {
        PRINTF("a non-correctable ECC event from system TCM \r\n");
    }
    if(peeloc == 8)
    {
        PRINTF("a 1 bit -correctable ECC event from code TCM \r\n");
    }
    if(peeloc == 9)
    {
        PRINTF("a 1 bit -correctable ECC event from system TCM \r\n");
    }
    if(peeloc == 14)
    {
        PRINTF("a PC Tag Parity Error \r\n");
    }
    if(peeloc == 15)
    {
        PRINTF("a PC Data Parity Error \r\n");
    }


    if(MCM->LMPEIR & (1<<9))
    {
        PRINTF("1 bit Error detected on system TCM \r\n");
    }
    if(MCM->LMPEIR & (1<<8))
    {
        PRINTF("1 bit Error detected on code TCM \r\n");
    }
    if(MCM->LMPEIR & (1<<1))
    {
        PRINTF("Non-correctable Error detected on system TCM \r\n");
    }
    if(MCM->LMPEIR & (1<<0))
    {
        PRINTF("Non-correctable Error detected on code TCM \r\n");
    }
}
```

```
void correctable_callback(void)
{
    PRINTF("correctable inerrupt happen. \r\n");
    decode_status();
    clear_status();
}
void uncorrectable_callback(void)
{
    PRINTF("uncorrectable inerrupt happen. \r\n");
    decode_status();
    clear_status();
    PRINTF("exit uncorrectable_callback \r\n");
}

void fix_error(void)
{
    int * addr = (int *)MCM->LMFAR;
    *addr = 0;
}


void dump_fault_info(void)
{
    PRINTF("MCM->LMFAR = %x \r\n",  MCM->LMFAR);
    PRINTF("MCM->LMFATR = %x \r\n", MCM->LMFATR);
    PRINTF("MCM->LMFDHR = %x \r\n", MCM->LMFDHR);
    PRINTF("MCM->LMFDLR = %x \r\n", MCM->LMFDLR);
}


int multi_bit_isr_cnt = 0;
int one_bit_isr_cnt = 0;

void NMI_Handler(void)
{
    PRINTF("----------  \r\n");
    PRINTF("NMI_Handler \r\n");
    dump_fault_info();
    fix_error();
    uncorrectable_callback();
    PRINTF("Exit NMI_Handler \r\n");
    multi_bit_isr_cnt++;
}
void CORE_IRQHandler(void)
{
    PRINTF("CORE_IRQHandler \r\n");
    dump_fault_info();
    correctable_callback();
    PRINTF("Exit CORE_IRQHandler. \r\n");
    one_bit_isr_cnt++;
}

void show_hardfault(void)
{
    PRINTF("hard fault. \r\n");
}
```

```
void HardFault_Handler(void)
{
    show_hardfault();
}

void test_ecc(void)
{
    volatile int v;
    volatile int * p = (int *)0x20000000;

    PRINTF("\r\n\r\n\r\n");
    PRINTF("Test 1: \r\n");
    v = *p;
    PRINTF("Done. \r\n");
    if((multi_bit_isr_cnt == 0) && (one_bit_isr_cnt == 0))
    {
        PRINTF("PASS. \r\n");
    }
    else
    {
        PRINTF("FAIL. \r\n");
        while(1)
            ;
    }
    PRINTF("\r\n\r\n\r\n");


    #define T_TEST 0x8000044C
    PRINTF("Test 2:  \r\n");
    enable_tcm_ecc();
    *p = T_TEST + 0; // 0x8000044C
    disable_tcm_ecc();
    *p = T_TEST + 1;
    enable_tcm_ecc();
    PRINTF("Read addr: %x\r\n", p);
    v = *p;
    PRINTF("Done.\r\n");
    PRINTF("v = %x\r\n", v);
    if((multi_bit_isr_cnt == 0) && (one_bit_isr_cnt == 1) && (v == T_TEST))
    {
        PRINTF("PASS. \r\n");
        one_bit_isr_cnt = 0;
    }
    else
    {
        PRINTF("FAIL. \r\n");
        while(1)
            ;
    }
    PRINTF("\r\n\r\n\r\n");

    PRINTF("Test 3:  \r\n");
    enable_tcm_ecc();
    *p = T_TEST + 0; // 0x8000044C
    disable_tcm_ecc();
    *p = T_TEST + 3;
    enable_tcm_ecc();
    PRINTF("Read addr: %x\r\n", p);
    v = *p;
    PRINTF("Done.\r\n");
```

```
    PRINTF("v = %x\r\n", v);
    if((multi_bit_isr_cnt == 1) && (one_bit_isr_cnt == 0))
    {
        PRINTF("PASS. \r\n");
        multi_bit_isr_cnt = 0;
    }
    else
    {
        PRINTF("FAIL. \r\n");
        while(1)
            ;
    }
    PRINTF("\r\n\r\n\r\n");
}

void preload_tcm(void)
{
    memset((void *)0x20000000, 0, 1024 * 128);
    memset((void *)0x1ffe0000, 0, 1024 * 128);
}

void dump_uuid(void)
{
    PRINTF("UUID:\r\n");
    PRINTF("%x: %x\r\n", &(OCOTP->FUSEN[16].FUSE), OCOTP->FUSEN[16].FUSE);
    PRINTF("%x: %x\r\n", &(OCOTP->FUSEN[17].FUSE), OCOTP->FUSEN[17].FUSE);
}


int main(void)
{
    char ch;

    /* Init board hardware. */
    BOARD_InitHardware();

    PRINTF("CM4 ECC validate test.\r\n");
    dump_uuid();
    EnableIRQ(CORE_IRQn);


    enable_tcm_ecc();
    clear_status();
    preload_tcm();
    test_ecc();

    while (1)
        ;
}
```

6 Acronyms

Table 1 lists the acronyms used in this document.

Table 1. Acronyms

Term	Description
ECC	Error-Correcting Code

Table 1. Acronyms...*continued*

Term	Description
TCM	Tightly-Coupled Memory

7 Note about the source code in the document

Example code shown in this document has the following copyright and BSD-3-Clause license:

Copyright 2026 NXP Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

1. Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
2. Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials must be provided with the distribution.
3. Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

8 Revision history

Table 2 summarizes the revisions to this document.

Table 2. Revision history

Document ID	Release date	Description
AN14960 v.2.0	11 September 2026	Initial public release
AN14960 v.1.0	16 March 2026	Initial NDA release

Legal information

Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at https://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this document expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

HTML publications — An HTML version, if available, of this document is provided as a courtesy. Definitive information is contained in the applicable document in PDF format. If there is a discrepancy between the HTML document and the PDF document, the PDF document has priority.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP B.V. — NXP B.V. is not an operating company and it does not distribute or sell products.

Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

Contents