NXP Semiconductors
Application Notes

Document Number: AN13067
Rev 0 11/2020

Flash ECC Error Report Path and Safety Mechanism Implementation

by: NXP Semiconductors

1. Introduction

MPC57XX family of 32-bit microcontrollers is widely used in integrated automotive application controllers. It belongs to an expanding range of automotive focused products designed for flexibility to support a variety of applications. The advanced and cost-efficient host processor core of the MPC57XX(includes MPC5746R, MPC5775B/E, MPC5777C) automotive controller family complies with the Power Architecture® embedded category. All devices in this family are built around a safety concept based on the delayed lockstep, targeting an ISO26262 ASIL-D (Design) integrity level.

To meet the higher level of security needs, the safety requirements from the chip safety manual were complemented and were integrated in the safety library. For more detailed information about MPC57XX Safety Library please refer to the safety manual.

This application note introduces some functions of the MPC57XX Safety Library. It focuses on introducing the theory and implementation of flash ECC error report path test.

Contents

2. ECC and Flash memory characters

2.1. MPC57XX flash memory

The primary function of the embedded flash memory is to serve as an electrically programmable and erasable non-volatile memory (NVM). It can be used for instruction and data storage. The following figure shows the top-level diagram and functional organization of the flash memory unit.

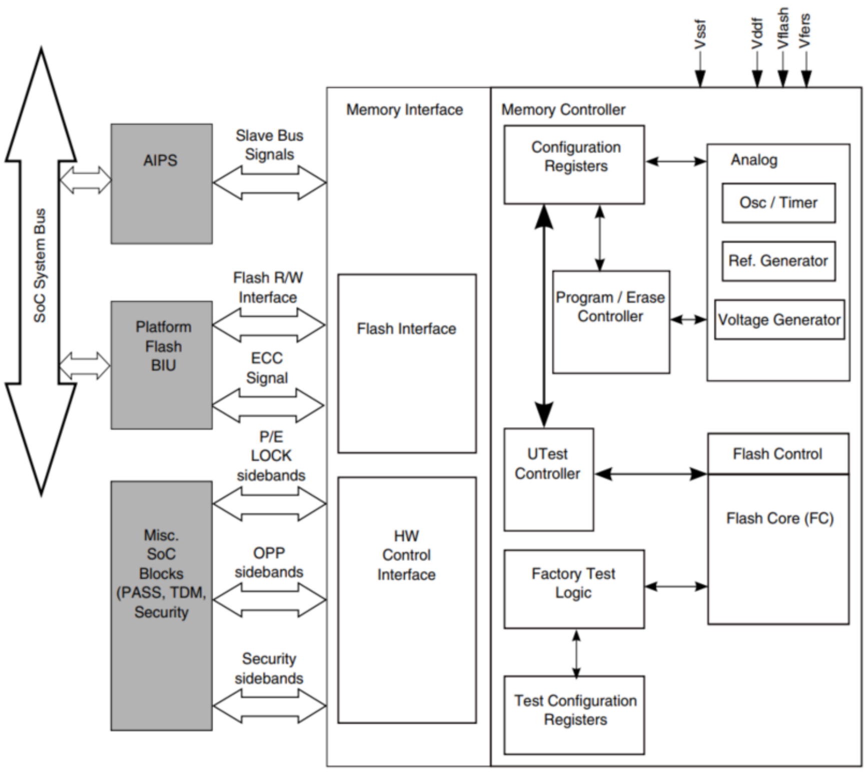


Figure 1. Block diagram

The embedded flash memory consists of address spaces in four groupings.

- Low address space
- Mid address space
- High address space
- 256 KB address space

The main address space is divided into partitions. Each address space mentioned above consists of a partition pair. Partitions are used to determine locations for valid read-while-write (RWW) operations. While the embedded flash memory is performing a write (program or erase) operation to a given partition, it can simultaneously perform a read operation for any other partition. For program operations, only the address specified by an interlock write determines the partition being written (block locking and block select registers do not determine the RWW partitions being written). For erase operations, only blocks that are selected and unlocked determine the RWW partitions being written.

The main address space is also divided into blocks to implement independent erase and program protection. The UTEST NVM block also exists as a block and has an independent program protection. The UTEST NVM block is included to support systems that require nonvolatile memory (NVM) for safety or to store system initialization information. Safety library shall use one block flash memory to execute the FLASH ECC error report path test. This block is also used to store safety information of other safety library test, for example, the MISRs of the Flash array integration test. This flash block shall be declared in the linker file and could not be used to store another data. And it also could not be the same partition with a flash of programming safety core binary.

The embedded flash memory is addressable by word (32 bits) or double word (64 bits) for program operation and page (256 bits) for a read operation. Multiple-words or double-word writes may be done in the flash memory to fill up the program page buffer (256bits), enabling page programming (256 bits, requiring 4 double-word writes) and quad-page programming (1024 bits, requiring 16 double-word writes). Flash memory reads always return 256 bits, although read page buffering may be performed by the Bus Interface Unit (BIU).

Flash memory on-chip consists of a flash memory controller and a flash memory array module. The flash controller provides flash configuration and control functions and manages the interface between the flash memory array and the device crossbar switch. The following figure shows the flash memory architecture. The remaining sections give the functional details of the flash controller and flash array, followed by the memory maps.

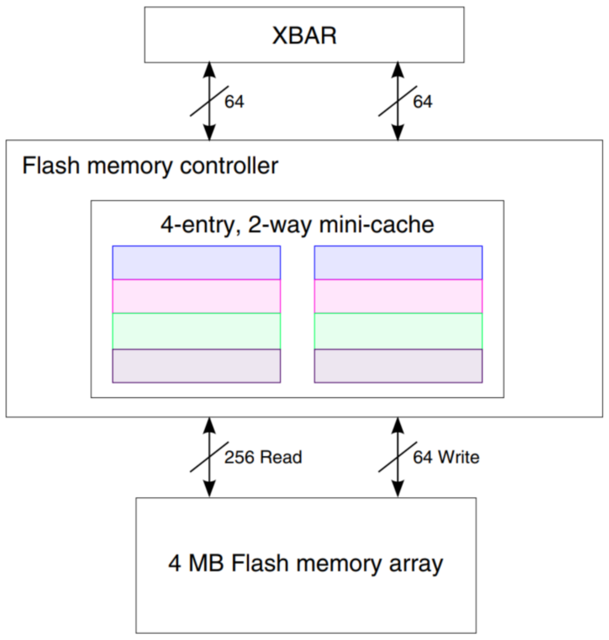


Figure 2. C55fmc memory block diagram

2.2. ECC introduction

To support market demand related to improved functional and transient fault detection capabilities, this family of automotive microcontrollers includes end-to-end ECC (e2eECC) support. This e2eECC is structurally different from traditional “ECC at memory” functionality because it provides robust error detection capabilities from one endpoint of an information transfer to another endpoint, with temporary information stored in one or more intermediate components.

Table 1. ECC comparison of data write-then-read sequence

Traditional ECC	End-to-End ECC (e2eECC)
Bus master initiates data write	Bus master initiates data write and generates ECC checkbits based on 29-bit address and 64-bit data fields for the computational shell, or based on 30-bit address and 32-bit data field for the peripheral shell
Data write transfer routed from bus master to appropriate bus slave	Data write transfer (including checkbits) routed from bus master to appropriate bus slave
For a bus memory slave, generate the ECC checkbits based on the data value and store data + checkbits into the memory	For a bus memory slave, store data and checkbits into the memory
Bus master initiates data read of previously written memory location	Bus master initiates data read of previously written memory location
Data read transfer routed from bus master to appropriate bus slave	Data read transfer routed from bus master to appropriate bus slave
For a bus memory slave, the memory array is accessed, the controller performs the ECC checkbit decode and syndrome generation, performs any needed single-bit correction and drives the read data onto the system bus interconnect	For a bus memory slave, the memory array is accessed, and the controller passes the read data and associated checkbits onto the system bus interconnection
The bus master captures the read data and continues	The bus master captures the read data and associated checkbits, performs the ECC checkbit decode and syndrome generation, performs any needed single-bit correction and continues

Memory protected by ECC/EDC traditionally generates and checks additional error parity information local to the memory unit to detect and/or correct errors that have occurred on data stored in the memory. On the other hand, e2eECC generates error protection codes at the source of data generation. It sends the encoded data and error protection codes to intermediate storage when a memory write is initiated by a bus master and performs a data integrity check using the previously-stored error protection codes at a data memory when a read of stored information is requested by a bus master. The intermediate storage may transform the generated error protection codes into another format for storage and then regenerate the codes for provision when a request is made to read the stored information or it may simply store the original protection codes unaltered, depending on the particular unit.

Additionally, the error protection codes are generated based on more than just the data associated with a storage location to protect additional information associated with access. In particular, address information corresponding to the access location of the stored information is combined with the stored data at the data source to generate error protection codes that cover certain types of addressing errors in the system interconnector or in the memory unit. The error protection codes may be checked at the memory unit on a store to ensure that no address information or data has been corrupted while the request has transitioned through the device from the bus master source. The error protection codes may also be checked to ensure that address decoding within the storage memory was performed properly (although not all address decoding errors can be detected this way) or it may simply store the received data and protection codes at the address it receives.

On a read request from a bus master, the memory unit retrieves the data information and error protection codes corresponding to the received address from the storage location or locations and supplies the data along with the error protection codes to the requesting device. The requesting device uses a locally stored address value corresponding to the read request to check the returning data and error protection codes to ensure that no errors have occurred in either addressing the memory or in the retrieved data. This ensures that the address sent for the request was not corrupted, the addressed location was actually accessed (to the extent it is possible to ensure) and the stored data was error-free, to the extent the ECC coding scheme can detect. As a result, the fault coverage provided by the end-to-end check is considerably more robust than the previous implementation of locally generated and checked/corrected error protection at each memory unit.

- Data check bits generation - external interfaces, 64-bit data ECC granularity
- Internal data check bits generation - 32-bit data ECC granularity
- The address portion of check bits generation, 64- or 32-bit data ECC granularity
- CACHE and IMEM data check bits generation - 64-bit ECC granularity
- D-CACHE and DMEM internal data check bits generation - 32-bit data ECC granularity

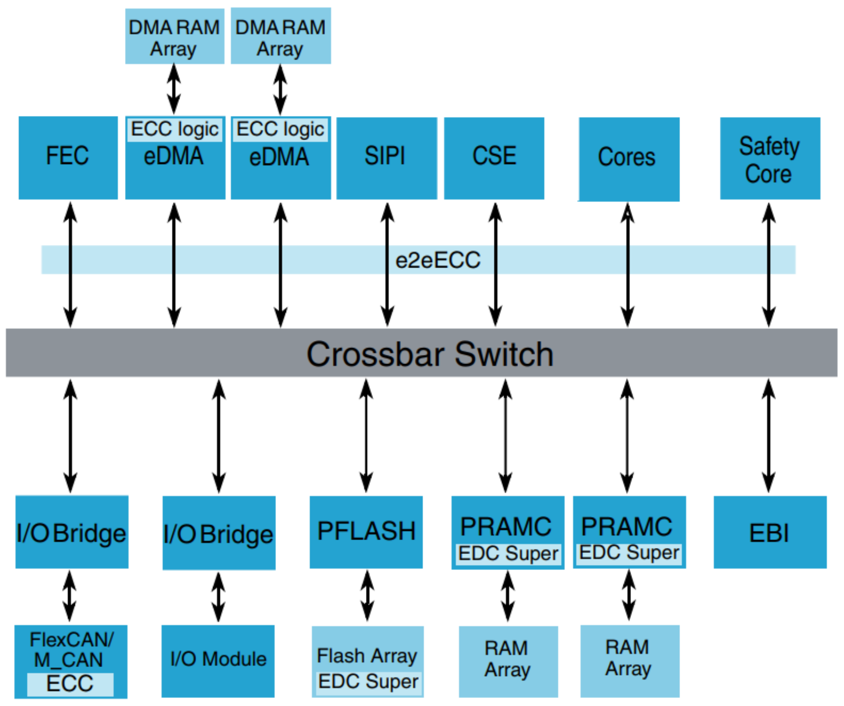


Figure 3. MPC5777C e2eECC overview

Single bit errors are automatically corrected and will be reported to the ERM/MEMU module. Multiple bit ECC errors will be detected and different platforms will cause different behavior.
For z0, z1, z3, and z6 versions of e200 cores.

Table 2. 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) *

For z4 and z7 versions of e200 cores.

Table 3. 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

3. ECC error injection implementation

3.1. ECC error injection

Flash ECC error injection test includes two parts, single-bit error injection test and multi-bit error injection test. The safety library uses one flash block(the last one of the large block in this demo) to store some safety information for the flash ECC test. This block is divided into three parts and allocation is shown in Figure 4 and Figure 5. The first part is used to store important information(eg: Flash ECC injection test header information). The second part and the third part are used for the flash ECC test. The second part is four-byte alignment, and the third part is eight-byte(8 bytes shall generate ECC) alignment. The second part is used to record the usage of this flash block and the third part is used for

ECC error injection. After completing an ECC error injection test, the second part shall be updated. The system will erase all this block when any one of the three blocks is full.

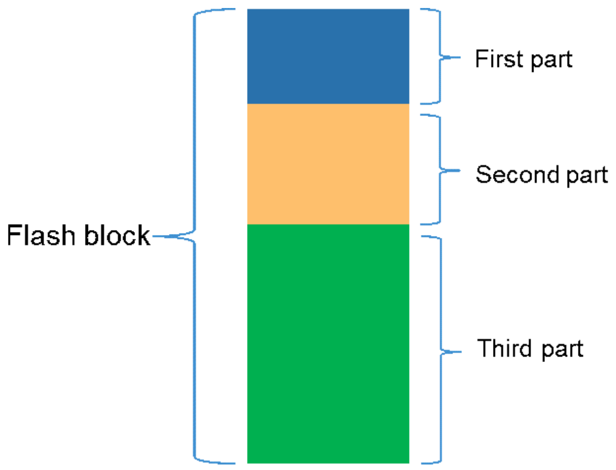


Figure 4. The flash block allocation

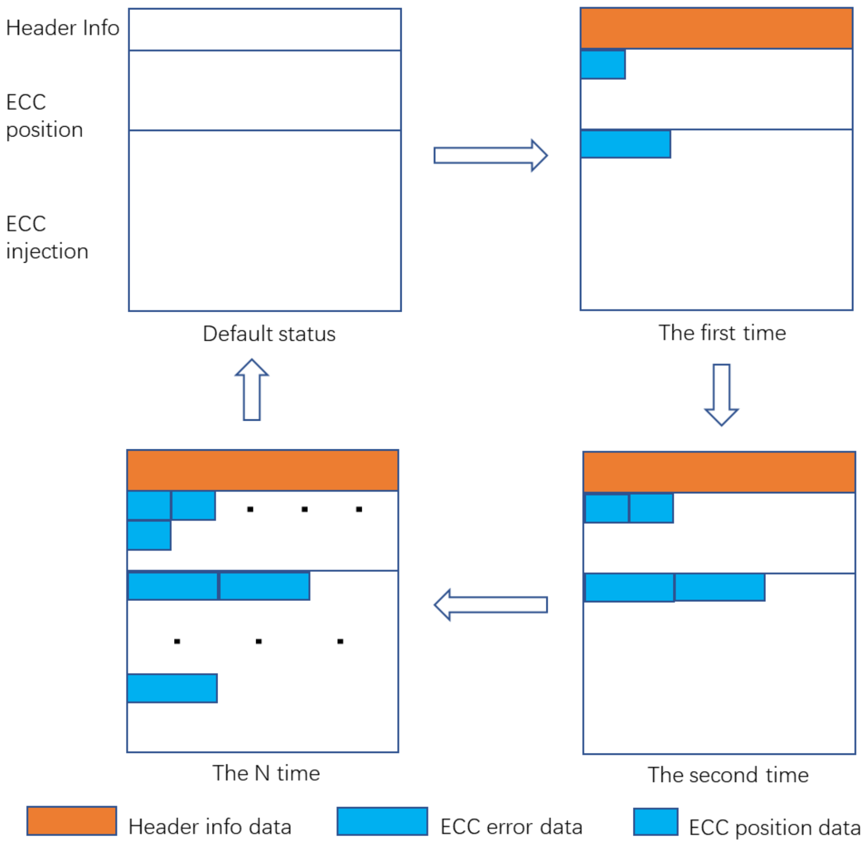


Figure 5. The ECC info store mechanism

3.1.1. Inject ECC 1-bit error

Flash single-bit ECC error needs to be programmed twice. Follow the steps for programming it:

1) Enable the single-bit ECC error report in the flash controller.

```
static void FS_FLASH_AccessEnableCEReport(const uint8_t flashInst)
{
    C55FMC_Type *pBase = s_pFlashRegList[flashInst];
    /* Enable single bit ECC error reporting in flash controller */
    pBase->UT0 = 0xF9F99999;    /* Enable UTE */
    /*Enable single bit error correction*/
    pBase->UT0 = C55FMC_UT0_SBCE(1);
    pBase->UT0 = C55FMC_UT0_UTE(0); /* Disable UTE */
}
```

2) Program to the same valid flash address twice to generate a single-bit ECC error. The first time program value are 0xFFFFFFFF and 0x00000000; and the second time program value are 0xFFFFFFFF and 0x00000001.

```
status = FS_FLASH_ProgramBytes(0xFFFFFFFF,  0x00000000, ...);
if (FS_STATUS_SUCCESS == status)
{
     status = FS_FLASH_ProgramBytes(0xFFFFFFFF, 0x00000001, ...)
}
```

3.1.2. Inject ECC 2-bits error

Flash 2-bits ECC error also needs to be programed twice. Program the same valid flash address twice to generate a 2-bits ECC error. Follow the below steps:

1) The first time program value: 0x00450000 and 0x00000000;

2) The second time program value: 0x00580000 and 0x00000000.

```
status = FS_FLASH_ProgramBytes(0x00450000, 0x00000000);
if  (FS_STATUS_SUCCESS == status)
{
     status = FS_FLASH_ProgramBytes(0x00580000, 0x00000000);
}
```

NOTE

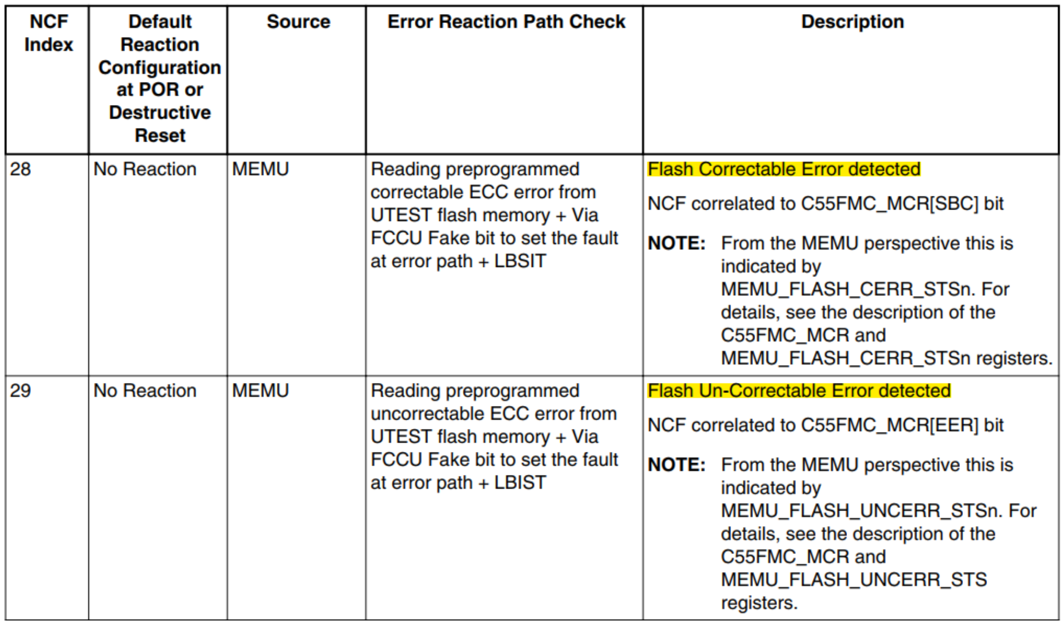
There are some special data that might generate the same ECC, so you need to avoid these cases. The data with the same ECC check bits is shown in Table 4.

Table 4. Data with the same ECC

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

3.2. ECC error detection

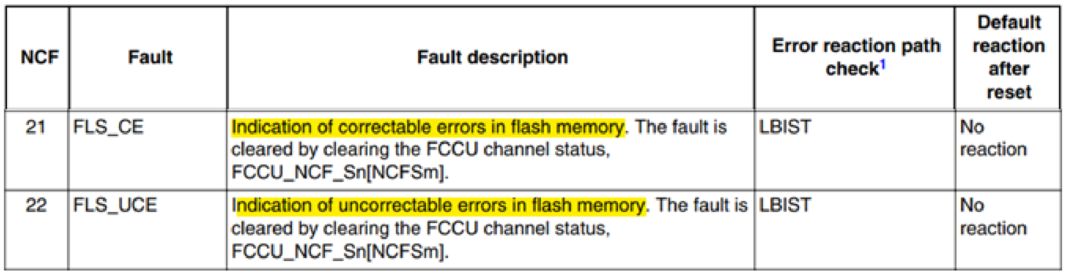
When a flash ECC error occurred, the error is reported to MEMU(MPC5746R)/ERM(MPC5775E, MPC5775B, MPC5777C) and FCCU module. MEMU module can record flash error type(1-bit / 2-bits) and error address. ERM module just record flash error type(1-bit / 2-bits).



NCF Index	Default Reaction Configuration at POR or Destructive Reset	Source	Error Reaction Path Check	Description
28	No Reaction	MEMU	Reading preprogrammed correctable ECC error from UTEST flash memory + Via FCCU Fake bit to set the fault at error path + LBSIT	Flash Correctable Error detected NCF correlated to C55FMC_MCR[SBC] bit **NOTE:** From the MEMU perspective this is indicated by MEMU_FLASH_CERR_STSn. For details, see the description of the C55FMC_MCR and MEMU_FLASH_CERR_STSn registers.
29	No Reaction	MEMU	Reading preprogrammed uncorrectable ECC error from UTEST flash memory + Via FCCU Fake bit to set the fault at error path + LBIST	Flash Un-Correctable Error detected NCF correlated to C55FMC_MCR[EER] bit **NOTE:** From the MEMU perspective this is indicated by MEMU_FLASH_UNCERR_STSn. For details, see the description of the C55FMC_MCR and MEMU_FLASH_UNCERR_STS registers.

Figure 6. MPC5746R FCCU channel for flash

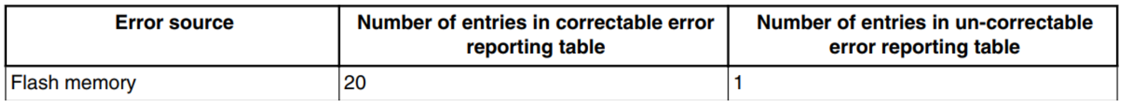
MPC5775E/ MPC5775B/MPC5777C FCCU channel for flash ECC error.



NCF	Fault	Fault description	Error reaction path check[1]	Default reaction after reset
21	FLS_CE	Indication of correctable errors in flash memory. The fault is cleared by clearing the FCCU channel status, FCCU_NCF_Sn[NCFSm].	LBIST	No reaction
22	FLS_UCE	Indication of uncorrectable errors in flash memory. The fault is cleared by clearing the FCCU channel status, FCCU_NCF_Sn[NCFSm].	LBIST	No reaction

Figure 7. MPC577x FCCU channel for flash

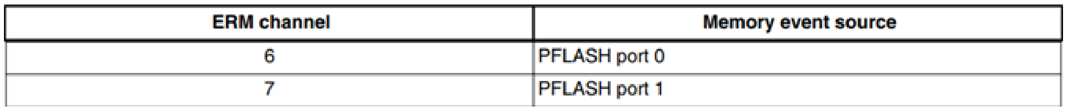
MPC5746R MEMU has 20 entries for flash 1-bit error and 1 entry for flash 2-bits error. Each entry in the reporting table corresponds to a unique error event.



Error source	Number of entries in correctable error reporting table	Number of entries in un-correctable error reporting table
Flash memory	20	1

Figure 8. MPC5746R MEMU entry for flash

MPC5775E/ MPC5775B/MPC5777C ERM channel for flash ECC error



ERM channel	Memory event source
6	PFLASH port 0
7	PFLASH port 1

Figure 9. MPC577x ERM channel for flash

3.3. Software Implementation

For software implementation follow the belowsteps:

1. Fetching the flash ECC injection address;

   FS_FLASH_CalcEccErrReportAddr(&singleBitAddr, &multiBitAddr);

2. Disable data cache;

   FS_CACHE_Disable(FS_CACHE_TYPE_DCACHE);

3. 2-bit flash ECC error will lead to IVOR1, need to install exception call back function to jump out IVOR1.

   FS_EXCEPTION_RegisterCallback(…);

3.3.1 ECC 1-bit software implementation

The software procedure(initialization, injection, and check error flag) is completed by the steps shown in the following figure.

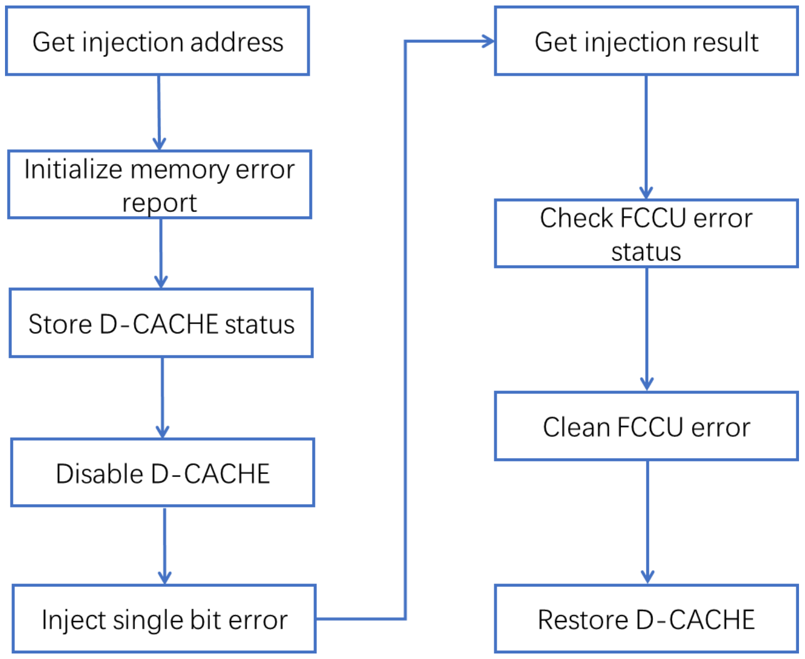


Figure 10. Flash 1-bit ECC error report path flow

3.3.2 ECC 2-bits software implementation

The software procedure(initialization, injection, and check error flag) is being done by the steps shown in the following figure.

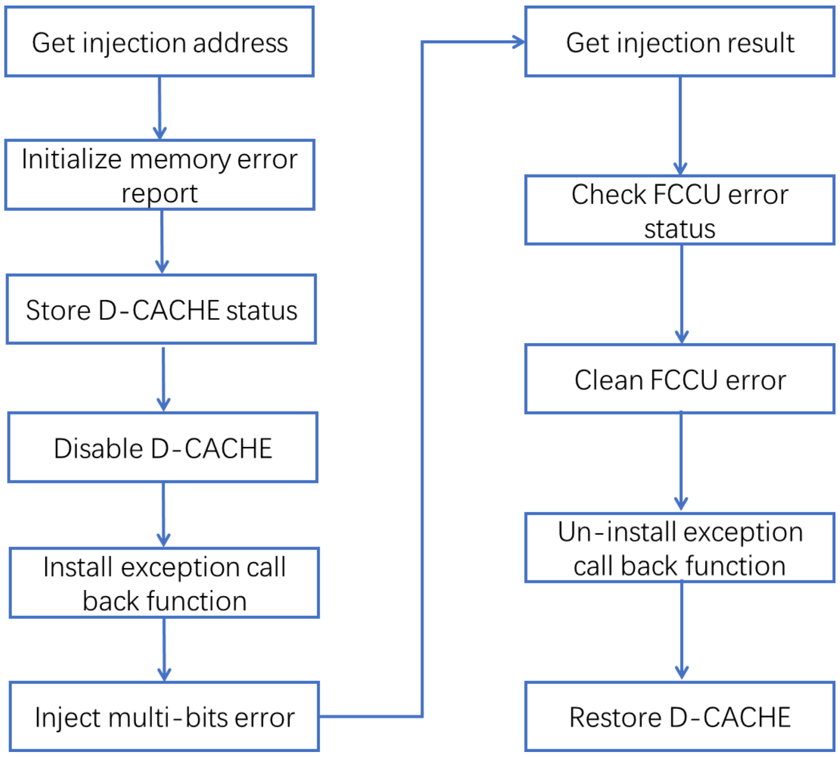


Figure 11. Flash 2-bits ECC error report path flow

4. Demo code and test results

The purpose of the demo is to show how to inject 1-bit and 2-bit ECC error in internal FLASH (you can choose the 1-bit or 2-bit operation depend on the special application) and how to check flash ECC error with safety library base on MPC57XX MCUs. The demo is only possible to run in the internal FLASH target and the test block is invalid for application. Ensure the UART is working normally because all the test information will be printed through it.

4.1. Test logic and code

The whole sequence (test initialization, injection, handling and print result) shown in the following steps:

1. Definition a test result variable for test.

```
fs_flash_ecc_report_path_test_result_t testObj;
```

2. Definition and initialize a test input variable for test.

```
fs_flash_ecc_report_path_test_param_t testCfg =
{
        .singleBitSelect = 1;
        .multiBitSelect = 1;
};
```

3. Configure the flash ECC test address in the linker file.

```
FS_FLASH_ECC_ERR_PATH_TEST_ADDR = 0xBC0000;
```

4. Set the PIT timer 0 as delay timer.

   FS_TIMER_DelayInit(0);

5. Initialize store safety information.

   FS_STORE_SafetyInfoStoreInit();

6. Initialize the flash ECC error report path test configuration.

   FS_FLASH_GetEccReportPathTestConfig(&testCfg);

7. Execute flash ECC error report path test.

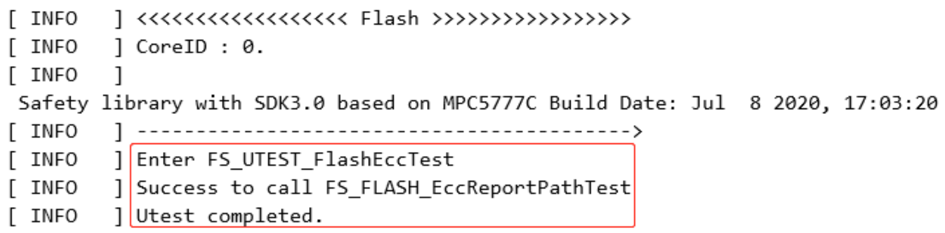
   FS_FLASH_EccReportPathTest(&testObj);

8. Check and print out test result.

```
if (!testObj.singleBitPassed)
{
        FS_PRINT_ERR("Failed to test Flash ECC single bit Err path.\n");
}
if (!testObj.multiBitPassed)
{
        FS_PRINT_ERR("Failed to test Flash ECC multi bit Err path.\n");
}
```

4.2. Test results

UART is used to print the result of the Flash ECC error report path test.



```
[ INFO   ] <<<<<<<<<<<<<<<<<< Flash >>>>>>>>>>>>>>>>>
[ INFO   ] CoreID : 0.
[ INFO   ]
 Safety library with SDK3.0 based on MPC5777C Build Date: Jul  8 2020, 17:03:20
[ INFO   ] ------------------------------------------>
[ INFO   ] Enter FS_UTEST_FlashEccTest
[ INFO   ] Success to call FS_FLASH_EccReportPathTest
[ INFO   ] Utest completed.
```

Figure 7. Flash ECC error report path test result

5. References

- MPC57XX PRM
- AN5288
- AN5200

How to Reach Us:

Home Page:
nxp.com

Web Support:
nxp.com/support

Document Number: AN13067
Rev 0
11/2020