

Introduction

This document describes the usage of IMA.

This design targets automotive applications and this cost-effective solution is based on the SPC582B60 device from STMicroelectronics.

Contents

- 1 Indirect Memory Access (IMA) 6**
 - 1.1 Introduction 6
 - 1.2 Accessing memory via the IMA module 6
 - 1.3 IMA module features 7
 - 1.4 IMA module block diagram 7
 - 1.5 IMA module memory map and registers 8

- 2 IMA module functional description 10**
 - 2.1 Unlocking read operations 10
 - 2.2 Unlocking write operations 10
 - 2.3 IMA read access 10
 - 2.4 IMA write access 11
 - 2.5 Application information 11

- 3 Hardware and software implementation 12**
 - 3.1 IMA testing DMA RAM ECC logic 12
 - 3.1.1 DMA RAM initialization 13
 - 3.1.2 Read operation by IMA on the selected memory 13
 - 3.1.3 Write operation by IMA on the selected memory inserting a SBE 13
 - 3.1.4 Read access by a master (e.g. CPU) 14
 - 3.1.5 Check relevant MEMU registers for SBE 14
 - 3.1.6 Write operation by IMA on the selected memory inserting a DBE 16
 - 3.1.7 Check relevant MEMU registers for DBE 16
 - 3.2 IMA testing CAN RAM ECC logic 17
 - 3.2.1 CAN RAM initialization 17
 - 3.2.2 Read operation by IMA on the selected memory 17
 - 3.2.3 Write operation by IMA on the selected memory inserting a SBE 18
 - 3.2.4 Read access by a master (e.g. CPU) 18
 - 3.2.5 Check relevant MEMU registers for SBE 19
 - 3.2.6 Write operation by IMA on the selected memory inserting a DBE 19
 - 3.2.7 Check relevant MEMU registers for DBE 19
 - 3.3 IMA testing SRAM (System RAM) ECC logic 20
 - 3.3.1 SRAM initialization 20

3.3.2	Read operation by IMA on the selected memory	20
3.3.3	Write operation by IMA on the selected memory inserting a SBE	21
3.3.4	Read access by a master (e.g. CPU)	21
3.3.5	Check relevant MEMU registers for SBE	21
3.3.6	Write operation by IMA on the selected memory inserting a DBE	22
3.3.7	Check relevant MEMU registers for DBE	22
4	Conclusion	23
5	Reference documents	24
6	Terms and Abbreviations	25
7	Revision history	26

List of tables

Table 1.	RAM arrays accessible via IMA	6
Table 2.	IMA module memory map.	9
Table 3.	Document revision history	26

List of figures

Figure 1. IMA Block diagram 8

1 Indirect Memory Access (IMA)

1.1 Introduction

Most of the volatile memories embedded in the microcontrollers of the SPC58 family are protected by an ECC. The hardware saves in the memory not only the data but also an ECC bits that depends on the data and - depending on the specific memory - on the address. During the normal functionality of the application, the software can't access the ECC bits.

The Indirect Memory Access (IMA) refers to an action that allows for accessing both data and relevant ECC bits in a volatile memory. This capability is important for testing activities, for example, verifying ECC functionality.

The SPC58 family MCU provides two mechanisms for IMA:

- The E2E ECC test capability ^(a) that provides a path for accessing data and ECC bits from some RAM arrays.
- An IMA controller module that provides registers for selecting, reading, and writing memory array including both data and relevant ECC bits.

1.2 Accessing memory via the IMA module

The IMA module provides an alternative way to access on-chip volatile memories without going through standard peripheral or system RAM interfaces. It can read or modify the entire content of the RAM including both data and relevant ECC bits.

The primary function of the IMA module is injecting an ECC error by providing invalid data or invalid ECC bits. Afterward, the user can access the faulty location to verify if the ECC logic can detect the error. For a correctable error, i.e. a single-bit error, the hardware corrects and reports the error. For a non-correctable error, i.e. a double-bit error, the peripheral reports the error.

The user can use this alternative path for diagnostic purposes by injecting a fault. For example, for RAM that does not support built-in E2E ECC, the IMA block provides a mechanism to write incorrect ECC values in the RAM. In such a way the user can verify the integrity of the ECC logic.

Not all RAM data and ECC bits are accessible via IMA module. The [Table 1](#) shows which RAM arrays are accessible via IMA module. See SPC582B60 reference manual ([Section 5: Reference documents](#)) for further details.

Table 1. RAM arrays accessible via IMA

IMA Array Selected	Block	Function	SPC582BX		
			Slot	Bits	ECC Bits
0	None	IMA disabled	-	-	-
1	Reserved	Reserved	-	-	-
3	DMA	DMA_1	64	72	71-64

a. This feature is built in the core. SPC582B60 e200z215An3 core does not support this feature.

Table 1. RAM arrays accessible via IMA (continued)

IMA Array Selected	Block	Function	SPC582BX		
			Slot	Bits	ECC Bits
3-8	Reserved	Reserved	-	-	-
9	CAN	CAN_0 Shared message RAM	1984	39	38-32
10-11	Reserved	Reserved	-	-	-
12	SRAM	SRAM_2_0_0	2048	40	39-32
13		SRAM_2_0_1	6144	40	39-32
14		SRAM_2_1	8192	40	39-32
15		SRAM_2_2	8192	40	39-32
16-31	Reserved	Reserved	-	-	-
32	CAN	CAN_1 Shared message RAM	1984	39	38-42
33-63	Reserved	Reserved	-	-	-

Please note the following restrictions on memory access via IMA:

- The user must unlock the IMA for read or write access.
- Once the user unlock the IMA and select a RAM for IMA accesses, no other accesses are allowed to this RAM until the IMA select line to the selected RAM is negated, i.e. no other accesses from any other master are expected until memory is deselected by the IMA.
- The FCC monitors the IMA when it accesses a location of a RAM. This enables the FCCU to monitor unwanted activation of the IMA. As a consequence, if the users need intentionally using the IMA, they must disable the respective FCCU channel before activating the IMA, otherwise, the FCCU triggers a fault.

1.3 IMA module features

The IMA module includes the following features:

- Enable reading and manipulation of data bits and ECC check bits.
- The following events occur in case a master accesses a memory selected by IMA:
 - Read operations return random data;
 - Write operations have no effect
- Any IMA access to memory are indicated in the FCCU
- Any IMA access to memory use a physical address, not a logical address, and thus memory addresses used by the IMA are not memory mapped.

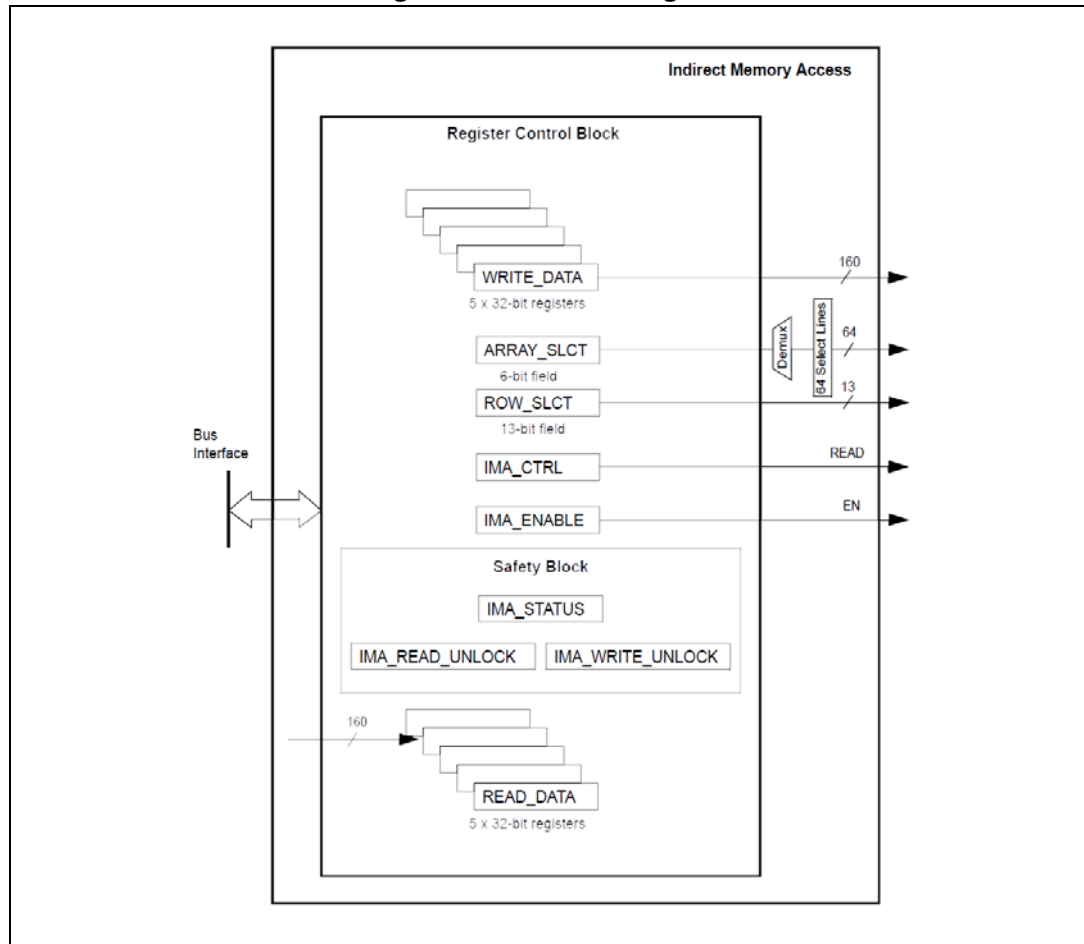
1.4 IMA module block diagram

The [Figure 1](#) shows the structure and function of the IMA module.

The Register Control Block contains the memory-mapped control registers accessible by software.

The Safety Block contains control registers to prevent spurious activation of the IMA module. It ensures the IMA does not access memory unless the system needs it too.

Figure 1. IMA Block diagram



1.5 IMA module memory map and registers

The [Table 2](#) provides details of the IMA module memory map.

See SPC582B60 reference manual ([Section 5: Reference documents](#)) for a detailed description of the registers in that map.

Table 2. IMA module memory map

Offset	Address register section
0x0000	IMA Control register (IMA_CTRL)
0x0004	IMA Enable Access register (IMA_ENABLE)
0x0008	IMA Status register (IMA_STATUS)
0x000C	IMA RAM SELECT register (IMA_SLCT)
0x0010	IMA Write Unlock register (IMA_WRITE_UNLOCK)
0x0014	IMA Read Unlock register (IMA_READ_UNLOCK)
0x0018-0x002B	Reserved
0x002C	IMA RAM Write Data register 4 (IMA_WRITE_DATA_4)
0x0030	IMA RAM Write Data register 3 (IMA_WRITE_DATA_3)
0x0034	IMA RAM Write Data register 2 (IMA_WRITE_DATA_2)
0x0038	IMA RAM Write Data register 1 (IMA_WRITE_DATA_1)
0x003C	IMA RAM Write Data register 0 (IMA_WRITE_DATA_0)
0x0040-0x004B	Reserved
0x004C	IMA RAM Read Data register 4 (IMA_READ_DATA_4)
0x0050	IMA RAM Read Data register 3 (IMA_READ_DATA_3)
0x0054	IMA RAM Read Data register 2 (IMA_READ_DATA_2)
0x0058	IMA RAM Read Data register 1 (IMA_READ_DATA_1)
0x005C	IMA RAM Read Data register 0 (IMA_READ_DATA_0)

2 IMA module functional description

This section describes the operation of the IMA from the unlocking of read and write accesses to the sequence to perform read or write accesses.

The IMA module control registers and logic are reset by hardware (system reset). A system reset de-asserts output signals and resets general configuration bits. See SPC582B60 reference manual ([Section 5: Reference documents](#)) for further details on IMA reset configuration.

2.1 Unlocking read operations

The IMA read access is not allowed unless it has been unlocked.

This is a safety feature to ensure the IMA does not access RAM unless the user intentionally does it.

To unlock the IMA read access, the software must consecutively write two static keys into the IMA_READ_UNLOCK.READ_KEY field:

- 1st write of key 0xF06AB5BC into the IMA_READ_UNLOCK.READ_KEY field.
- 2nd write of key 0x14081B56 into the IMA_READ_UNLOCK.READ_KEY field.

The IMA module has two possibilities to lock the IMA read accesses:

- writing an incorrect key into the IMA_READ_UNLOCK.READ_KEY field,
- not setting the IMA_ENABLE.EN bit for 16384 system clocks.

2.2 Unlocking write operations

The IMA write access will not be allowed unless it has been unlocked.

This is a safety feature to ensure the IMA does not access RAM unless the user intentionally does it.

To unlock the IMA read access, the software must consecutively write two static keys into the IMA_WRITE_UNLOCK.WRITE_KEY:

- 1st write of key 0x04A43F95 into the IMA_WRITE_UNLOCK.WRITE_KEY field.
- 2nd write of key 0xE4A9EBF7 into the IMA_WRITE_UNLOCK.WRITE_KEY field.

The IMA module has two possibilities to lock the IMA write accesses:

- writing an incorrect keyword into the IMA_WRITE_UNLOCK.WRITE_KEY field,
- not setting the IMA_ENABLE.EN bit for 16384 system clocks.

2.3 IMA read access

In order to perform an IMA read access, the user must follow the sequence below:

1. Unlock IMA read access by performing the two appropriate consecutive write operations into the IMA_READ_UNLOCK.READE_KEY field;
2. Based on the memory address to be accessed, write the IMA_SLCT.ARRAY_SLCT and IMA_SLCT.ROW_SLCT fields;
3. Set the IMA_CTRL.READ bit;
4. Set IMA_ENABLE.EN bit to initiate read access. This bit will self-clear after two clocks;
5. Wait an appropriate number of clocks depending on the speed of the selected memory;
6. Read data from IMA_READ_DATA registers. This causes data from RAM to be latched into the IMA and returned to the CPU on the next clock.

2.4 IMA write access

In order to perform an IMA read access the user must follow the sequence below:

1. Unlock IMA write access by performing the two appropriate consecutive write operations into the IMA_WRITE_UNLOCK.WRITE_KEY field;
2. Based on the memory address to be accessed, write the IMA_SLCT.ARRAY_SLCT and IMA_SLCT.ROW_SLCT fields;
3. Write the data in IMA_WRITE_DATA registers;
4. Clear the IMA_CTRL.READ bit;
5. Set IMA_ENABLE.EN bit to initiate write access. This bit will self-clear after two clocks;
6. Wait an appropriate number of clocks depending on the speed of the selected memory

2.5 Application information

Warning: the IMA inhibits correct operation of the underlying peripherals. The application software is responsible for ensuring there is not a resource conflict.

The user needs to configure portions of the SoC prior to using IMA module. The configuration depends on the particular application. The IMA can cause problems with the system or the peripheral if both the IMA and the other system components access the RAM during normal operation.

Please consult SPC582B60 reference manual ([Section 5: Reference documents](#)) to learn how to put the peripheral into a quiet state so that it will not access the RAM during the IMA access.

3 Hardware and software implementation

The aim of the document is to describe how to use the IMA module to program invalid data or invalid ECC bits and see if the ECC logic of the selected memory is able to detect the injected error.

For a correctable error, i.e. a single-bit error, the ECC logic of the selected memory corrects and reports the error; for a non-correctable error, i.e. a double-bits error, the ECC logic of the selected memory just reports it without any correction.

This document uses the SPC582B60 and its SPC58XXADPT64S Evaluation Board see SPC582B60 reference manual ([Section 5: Reference documents](#)) and SPC58XXADPT64S Data brief ([Section 5: Reference documents](#)) for further details. The user can utilize these concepts for all other devices belonging to the SPC58 family.

3.1 IMA testing DMA RAM ECC logic

According to the [Table 1](#) the IMA module is able to perform a 72-bits access (read or write operation) on DMA RAM memory. The first 64-bits are the data to be read or to be written in DMA RAM memory, and the following 8-bits are the ECC of the data.

In case of read operations, the IMA module load the 72-bits into the registers IMA_READ_DATA_x (with x = 0,1,2). The registers IMA_READ_DATA_0 and IMA_READ_DATA_1 contain the data and the register IMA_READ_DATA_2 contains the ECC.

Note that the 2nd data word is stored in the register IMA_READ_DATA_0 and the 1st data word is stored in the register IMA_READ_DATA_1. Moreover, the registers IMA_READ_DATA_x (with x = 3,4) are not used in this case.

In a similar way, in case of write operation, the IMA module stores the 72-bits to the registers IMA_WRITE_DATA_x (with x = 0,1,2). The register IMA_WRITE_DATA_0 contains the 2nd data word to be written, the register IMA_WRITE_DATA_1 contains the 1st data word to be written, and the register IMA_WRITE_DATA_2 contains the ECC to be written. As a result, the IMA module can program invalid data or invalid ECC bits. The user can utilize this method to verify the integrity of the ECC logic.

SPC582B60 device has a DMA with 16 TCD channels and the size of each TCD channel is 32-bytes. As result, the total size of DMA RAM is 512-bytes. Considering the 64-bits data access for DMA, they correspond to 64 IMA slots as listed in the [Table 1](#). Moreover, considering the memory map of the SPC582B60 device and the memory map of the DMA IP, the DMA RAM is located to the address range 0xF40A_5000-0xF40A_51FF.

In order to verify the integrity of the ECC logic that protects the RAM of the DMA, the user can follow the following steps²:

1. DMA RAM is initialized with known data;
2. Read operation by IMA is performed on the selected memory location;
3. Write operation by IMA is performed on the selected memory location injecting an error;
4. Read access by a master (e.g. CPU) is performed on the selected memory location;
5. Check relevant MEMU registers to verify the correct detection of the error.

3.1.1 DMA RAM initialization

In this example, the DMA RAM is initialized with data 0x000000xx (with xx = 0x01,...,0x80) starting from the address 0xF40A_5000 to the address 0xF40A_51FC. This step can be skipped if the DMA RAM (or the selected part of DMA RAM) is already initialized with known data.

3.1.2 Read operation by IMA on the selected memory

This example uses as reference the memory location at address 0xF40A_5008.

IMA read access is unlocked by writing the two key in the IMA_READ_UNLOCK.READ_KEY field (first IMA_READ_UNLOCK.READ_KEY = 0xF06A_B5BC and then IMA_READ_UNLOCK.READ_KEY = 0x1408_1B56). The user can verify the correct unlocking by checking that the relevant status bit in the IMA status register is set (IMA_STATUS.READ_LOCK = 1).

After unlocking the IMA read access, the memory location to be tested is selected using the fields IMA_SLCT.ARRAY_SLCT (IMA_SLCT.ARRAY_SLCT = 2 in order to select DMA RAM), IMA_SLCT.ROW_SLCT (IMA_SLCT.ROW_SLCT = 1 in order to select the 2nd slot starting at address 0xF40A_5008), and the IMA_CTRL.READ (IMA_CTRL.READ = 1).

The user starts the read operation by setting the IMA_ENABLE.EN bit (IMA_ENABLE.EN = 1). This bit is a self-clear bit after two clocks. The read operation ends when hardware clears this bit.

The read data are available in the registers IMA_READ_DATA_x (with x = 0,1,2). In this example test, IMA_READ_DATA_0 = 0x00000004, IMA_READ_DATA_1 = 0x00000003 and IMA_READ_DATA_2 = 0x000000AF. 0xAF is the 8-bits ECC corresponding to these for these data and address.^(b)

Note that the computation of the 8-bits ECC includes only the base offset 0x08 of target address 0xF40A5008.

Finally, in order to allow a normal access from a master (e.g. CPU), the user must de-select the memory selection (IMA_SLCT.ARRAY_SLCT = 0 and IMA_SLCT.ROW_SLCT = 0).

3.1.3 Write operation by IMA on the selected memory inserting a SBE

To insert a single bit error in the memory location at the address 0xF40A_5008, IMA write access is unlocked writing the two unique key in the IMA_WRITE_UNLOCK.WRITE_KEY field (first IMA_WRITE_UNLOCK.WRITE_KEY = 0x04A4_3F95 and then IMA_WRITE_UNLOCK.WRITE_KEY = 0xE4A9_EBF7). User can verify the correct unlocking by checking that the relevant status bit in the IMA status register IMA_STATUS is set (IMA_STATUS.WRITE_LOCK = 1).

After unlocking the IMA write access, the memory location to be tested is selected using the fields IMA_SLCT.ARRAY_SLCT (IMA_SLCT.ARRAY_SLCT = 2 in order to select DMA RAM), IMA_SLCT.ROW_SLCT (IMA_SLCT.ROW_SLCT = 1 in order to select the 2nd slot

b. Not all peripheral memories include the address in the ECC algorithm. Refer to chapter "Functional safety" of the SPC582B60 Reference Manual (1) to have all details

starting at address 0xF40A_5008), and the IMA_CTRL.READ bit is cleared (IMA_CTRL.READ = 0).

In this example test, a single bit error, i.e. a correctable error, is inserted in the 62th position of the data using the mask 0x40000000_00000000 for the read data double-word and the mask 0x000000000 for the read ECC word and writing the registers IMA_WRITE_DATA_x (with x = 0,1,2) with the values XOR-ed with proper mask:

- IMA_WRITE_DATA_0 = IMA_READ_DATA_0^0x00000000
- IMA_WRITE_DATA_1 = IMA_READ_DATA_1^0x40000000
- IMA_WRITE_DATA_2 = IMA_READ_DATA_2^0x00000000

As a result

- IMA_WRITE_DATA_0 = 0x00000004
- IMA_WRITE_DATA_1 = 0x00000003 and
- IMA_WRITE_DATA_2 = 0x000000AF

The registers IMA_WRITE_DATA_x (with x = 3,4) are not used in this case.

The user starts the write operation by setting the IMA_ENABLE.EN bit (IMA_ENABLE.EN = 1). This bit is a self-clear bit after two clocks. The write operation ends when the hardware clears this bit.

Finally, in order to allow access to by other master (e.g. CPU), the user must de-select the memory location (IMA_SLCT.ARRAY_SLCT = 0 and IMA_SLCT.ROW_SLCT = 0).

3.1.4 Read access by a master (e.g. CPU)

After the insertion of the error, a master issues a read access to the selected memory location in order to read the corrupted data, and detect the injected single bit error.^(c)

3.1.5 Check relevant MEMU registers for SBE

The MEMU collects and reports error events detected by the ECC logic that monitors the system RAM, peripheral RAM and FLASH memory.

When an ECC error event occurs, the MEMU records the cause of the event ^(d)and forwards the event to the FCCU. The ECC logic sends to the MEMU the address of the corrupted location and - in case of correctable error - the error syndrome ^(e)indicating the position of the faulty bit.

To verify the integrity of the ECC logic, the user must verify if the MEMU logs the correct address and bad bit. In this example, the injected single bit error is detected and reported to the MEMU. Indeed, the valid bit (PERIPH_RAM_CERR_STS0.VLD = 1) indicates that the

c. In this example, the selected memory location is read using a dummy variable and a properly initialized memory pointer (PTR = 0xF40A5000 and DUMMY_VAR = *(PTR+2)).

d. E.g. correctable error in the system RAM.

e. The MEMU doesn't record the syndrome for all memories. Refer to the SPC582B60 Reference Manual (1) to have all details.

entry in the MEMU reporting table is valid. In addition, this entry reports the address and the position of the faulty bit:

- PERIPH_RAM_CERR_STS0.BAD_BIT = 0x4A, and
- PERIPH_RAM_CERR_ADDR0 = 0xF40A_5008

The assertion of the valid bit causes the assertion of the correctable error detect flag (ERR_FLAG.PR_CE = 1).

Note that the syndrome (i.e. the BAD_BIT_value) does not indicate directly the position of the faulty bit but it is connected with this position. In fact, the reported syndrome is the ECC of the mask 0x40000000_00000000 used to modify the read data double-word and it indicates a single bit error in the 62th position of the data double-word. As result, in order to know the position of the faulty bit, the user needs to know the ECC of all possible mask and check which of these matches the syndrome.

Anyway, in this example, in order to check the correct indication of the single bit error position, the user can calculate (or read by IMA) the ECC of the address corresponding to the ECC for the selected memory location 0xF40A_5008 filled with all-zero data 0x00000000_00000000 (in this example it is 0x2F) and the ECC for the selected memory location 0xF40A_5008 filled with mask 0x40000000_00000000 (in this example it is 0x65) and then verify that the syndrome is equal to the address ECC (memory location filled with all-zero data) XOR-ed with the ECC for the selected memory location 0xF40A_5008 filled with mask 0x40000000_00000000 (in this example $0x65 \oplus 0x2F = 0x4A$.)

To summarize, to verify the capability of the ECC logic to detect a single bit error the user must:

1. write a random value to the target address (e.g. 0xF40A_5008) by a master (e.g. core)
2. read both data and ECC bits from the target address by the IMA
3. modify the data by XOR-ing it with the mask (e.g. 0x40000000_00000000);
4. write back the modified data and ECC bits to the target address by the IMA
5. read the target address by a master (e.g. core) to trigger the detection and correction
6. read the MEMU status registers to verify the correct reporting, i.e. the reported address is the target address and the syndrome is the ECC of the mask.

The ECC values used for the syndrome check (e.g. 0x2F and 0x65) can be calculated using the ECC algorithm or can be read by IMA filling the selected memory location before with all-zero data and then with the mask.

In case of single bit error, i.e. correctable error, in a certain memory location, the error is detected and the data is corrected by ECC logic before it is provided to the requesting master. As result, the memory location still holds corrupted data, as can be verified performing a new read operation by IMA on the same memory location.

Note that a new read access by a master, e.g. CPU, on the same memory location generates a new single bit error detection (and the correction of the data provided to the requesting master) with the reporting to the MEMU but, in this case, the MEMU will drop the error because the address and the syndrome are the same of the error already recorded in the MEMU reporting table. On the contrary, if the valid bit PERIPH_RAM_CERR_STS0.VLD was cleared before, the error is recorded again.

3.1.6 Write operation by IMA on the selected memory inserting a DBE

To insert a double bit error in the memory location at the address 0xF40A_5008, the IMA write access is unlocked writing the two unique key in the IMA_WRITE_UNLOCK.READ_KEY field (first IMA_WRITE_UNLOCK.READ_KEY = 0x04A4_3F95 and then IMA_WRITE_UNLOCK.READ_KEY = 0xE4A9_EBF7). The user can verify the correct unlocking operation by checking that the relevant status bit in the IMA status register IMA_STATUS is set (IMA_STATUS.WRITE_LOCK = 1).

After unlocking the IMA write access, the users select the memory location using the fields IMA_SLCT.ARRAY_SLCT (IMA_SLCT.ARRAY_SLCT = 2 to select DMA TCD storage), IMA_SLCT.ROW_SLCT (IMA_SLCT.ROW_SLCT = 1 to select the 2nd slot starting at address 0xF40A_5008), and the IMA_CTRL.READ bit is cleared (IMA_CTRL.READ = 0).

In this example test, considering that the selected memory location already contains corrupted data, i.e. a single bit error in the 62th position of the data double word, a 2nd single bit error is inserted in the 60th position of the data double-word and the mask 0x100000000_00000000 for the read data double-word and the mask 0x000000000 for the read ECC word ^(f)The registers IMA_WRITE_DATA_x (with x = 0,1,2) are filled with the values read by IMA and XOR-ed with proper masks IMA_WRITE_DATA_0 = IMA_READ_DATA_0^0x00000000, IMA_WRITE_DATA_1 = IMA_READ_DATA_1^0x10000000 and IMA_WRITE_DATA_2^0x00000000).

As result, in this example test, a double bit error, i.e. an uncorrectable error, is inserted in the selected memory location (IMA_WRITE_DATA_0 = 0x00000004, IMA_WRITE_DATA_1 = 0x50000003 and IMA_WRITE_DATA_2 = 0x000000AF with registers IMA_WRITE_DATA_x (with x = 3,4) not used). Note that the double bit error could be inserted in a different memory location holding not-corrupted data or the content of the selected memory location 0xF40A_5008 could be re-written by a master, e.g. CPU, before the insertion of a double bit error by IMA. In this case the proper masks to be used are 0x50000000_00000000 for the read data double-word and 0x000000000 for the read ECC word.

The user starts the write operation by setting the IMA_ENABLE.EN bit (IMA_ENABLE.EN = 1). This bit is a self-clear bit after two clocks. When the hardware clears this bit the write operation ends.

To allow access the fault location by a master (e.g. CPU), the user must de-select the memory location (IMA_SLCT.ARRAY_SLCT = 0 IMA_SLCT.ROW_SLCT = 0) and perform the read access, e.g. CPU, to the selected memory location in order to read the corrupted data and detect the injected double bit error.

3.1.7 Check relevant MEMU registers for DBE

In this example test, the inserted double bit error is correctly detected and reported to the MEMU, in fact the valid bit is asserted (PERIPH_RAM_UNCERR_STS.VLD = 1) indicating that the entry in the MEMU reporting table is valid (PERIPH_RAM_UNCERR_ADDR = 0xF40A5008). The assertion of the valid bit causes the uncorrectable error detect flag to be asserted (ERR_FLAG.PR_UCE = 1).

f. In this example, a double bit error is inserted by the injection of two consecutive single bit errors. It's possible to insert directly a double bit error using the proper mask.

Note that in case of uncorrectable error triggered by a read access done by the CPU, the hardware issues a machine check exception to the core.

3.2 IMA testing CAN RAM ECC logic

According to the [Table 1](#), IMA is able to perform a 39-bits access in the CAN RAM memory. The first 32-bits are the data and the remaining 7-bits are the ECC bits.

In case of read operations using the IMA, the IMA module stores the read 39-bits in the registers IMA_READ_DATA_x (with x = 0,1). The register IMA_READ_DATA_0 contains data and the register IMA_READ_DATA_1 contains the 7-bits ECC. The registers IMA_READ_DATA_x (with x = 2,3,4) are not used.

In a similar way, in case of write operation, IMA module stores the 39-bits to be written in the registers IMA_WRITE_DATA_x (with x = 0,1). The register IMA_WRITE_DATA_0 contains data, and IMA_WRITE_DATA_1 contains the 7-bits ECC. As result, the IMA module can save invalid data or invalid ECC bits. Using this feature the user can verify the integrity of the ECC logic of the selected peripheral.

SPC582B60 device has two CAN with 7-KBytes shared message RAM. As result, the total size of each CAN RAM corresponds, considering the 32-bits data access for the CAN, to the 1984 IMA slots as listed in the [Table 1](#). Moreover, considering the memory map of the SPC582B60 device and the memory map of the CAN peripherals, the CAN subsystem 0 RAM memory is located to the address range 0xF7ED4000-0xF7ED5EFF and the CAN subsystem 1 RAM memory is located to the address range 0xFBED4000-0xFBED5EFF and the CAN subsystem 1 RAM memory is located to the address range 0xFBED4000-0xFBED5EFF.

In order to perform the test, the user can follow the following steps:

1. The CAN RAM is initialized with known data;
2. Read operation by IMA is performed on the selected memory location;
3. Write operation by IMA is performed on the selected memory location injecting an error;
4. Read access by a master (e.g. CPU) is performed on the selected memory location;
5. Check relevant MEMU registers to verify the correct detection of the error.

3.2.1 CAN RAM initialization

In this example, the CAN RAM is initialized with data 0x00000xxx (with xxx = 0x001,...,0x7C0) starting from the address 0xF7ED_4000 to the address 0xF7ED_5EFC for CAN subsystem 0 RAM memory and starting from the address 0xFBED_4000 to the address 0xFBED_5EFF for CAN subsystem 1 RAM memory. The user can skip this step if the CAN RAM (or the selected part of CAN RAM) has been already initialized with known data.

3.2.2 Read operation by IMA on the selected memory

In this example test, the memory locations at the address 0xF7ED4008 and 0xFBED4008 are tested.

After unlocking the IMA read accesses,⁽⁹⁾ the user selects the memory location to be tested using the fields IMA_SLCT.ARRAY_SLCT (IMA_SLCT.ARRAY_SLCT = 9 in order to select CAN subsystem 0 RAM memory or IMA_SLCT.ARRAY_SLCT = 32 in order to select CAN

subsystem 1 RAM memory), IMA_SLCT.ROW_SLCT (IMA_SLCT.ROW_SLCT = 2 in order to select the 3rd slot starting at address 0xF7ED_4008 for CAN subsystem 0 or at address 0xFBED_4008 for CAN subsystem 1), and the IMA_CTRL.READ bit is set (IMA_CTRL.READ = 1).

The user starts the read operation by setting the IMA_ENABLE.EN bit. This bit is a self-clear bit after two clocks. When the hardware clears this bit the read operation ends.

The read data are available in the registers IMA_READ_DATA_x (with x = 0,1). In this example, IMA_READ_DATA_0 = 0x00000003 and IMA_READ_DATA_1 = 0x00000047^(h)

Note that for the CAN RAM, the computation of the 7-bits ECC does not include the target address.

Finally, in order to allow access to by other master (e.g. CPU), the user must de-select the memory location (IMA_SLCT.ARRAY_SLCT = 0 IMA2740_SLCT.ROW_SLCT = 0).

3.2.3 Write operation by IMA on the selected memory inserting a SBE

In order to insert a single bit error in the memory location at the address 0xF7ED_4008 or 0xFBED_4008, after the unlocking of the IMA write access⁷, the user selects the memory location by the fields IMA_SLCT.ARRAY_SLCT (IMA_SLCT.ARRAY_SLCT = 9 in order to select CAN subsystem 0 RAM memory or IMA_SLCT.ARRAY_SLCT = 32 in order to select CAN subsystem 1 RAM memory), IMA_SLCT.ROW_SLCT (IMA_SLCT.ROW_SLCT = 2 in order to select the 3rd slot starting at address 0xF7ED_4008 for CAN subsystem 0 or at address 0xFBED_4008 for CAN subsystem 1), and the IMA_CTRL.READ bit is cleared (IMA_CTRL.READ = 0).

In this example, a single bit error is inserted in the 32th position of the data using the mask 0x80000000 for the read data word and the mask 0x00000000 for the read ECC word:

- IMA_WRITE_DATA_0 = IMA_READ_DATA_0 ^ 0x80000000, and
- IMA_WRITE_DATA_1 = IMA_READ_DATA_1 ^ 0x00000000

As a result, in this example test, IMA_WRITE_DATA_0 = 0x80000003 and IMA_WRITE_DATA_1 = 0x00000047. The registers IMA_WRITE_DATA_x (with x = 2,3,4) are not used in this case.

The user starts the write operation by setting the IMA_ENABLE.EN bit. This bit is a self-clear bit after two clocks. When the hardware clears this flag the write operation ends.

Finally, to allow access to by other master (e.g. CPU), the user must de-select the memory location (IMA_SLCT.ARRAY_SLCT = 0 IMA_SLCT.ROW_SLCT = 0).

3.2.4 Read access by a master (e.g. CPU)

After the injection of the error, a read access to the selected memory location is performed to read the corrupted data and detect the injected single bit error.

g. See section 2.3 and 3.1.2 for further details.

h. 0x47 is 7-bits ECC for this data.

3.2.5 Check relevant MEMU registers for SBE

In this example, the injected single bit error is correctly detected and reported to the MEMU. Indeed, the valid bit is asserted (`PERIPH_RAM_CERR_STS0.VLD = 1`) indicating that the entry in the MEMU reporting table is valid (`PERIPH_RAM_CERR_ADDR0 = 0xF7ED_4008` or `0xFBED_4008` and `PERIPH_RAM_CERR_STS0.BAD_BIT = 0xFF`). The assertion of the valid bit causes the correctable error detect flag to be asserted (`ERR_FLAG.PR_CE = 1`).

Note that in this case, i.e. for CAN RAM, the syndrome does not indicate the bit position of the detected error. It is always read as `0xFF`.

In case of single bit error, i.e. correctable error, in a certain memory location, the error is detected and the data is corrected by ECC logic before it is provided to the requesting master. As result, the memory location still holds corrupted data, as can be verified performing a read operation by IMA on the memory location.

3.2.6 Write operation by IMA on the selected memory inserting a DBE

To insert a double bit error in the memory location at the address `0xF7ED_400C` or `0xFBED_400C`, after the unlocking of the IMA write access, the memory location to be tested is selected using the fields `IMA_SLCT.ARRAY_SLCT` (`IMA_SLCT.ARRAY_SLCT = 9` in order to select CAN subsystem 0 RAM memory or `IMA_SLCT.ARRAY_SLCT = 32` in order to select CAN subsystem 1 RAM memory), `IMA_SLCT.ROW_SLCT` (`IMA_SLCT.ROW_SLCT = 3` in order to select the 4th slot starting at address `0xF7ED_400C` for CAN subsystem 0 or at address `0xFBED_400C` for CAN subsystem 1), and the `IMA_CTRL.READ` bit is cleared (`IMA_CTRL.READ = 0`).

In this example test, a double bit error, i.e. an uncorrectable error, is inserted by using the mask `0x81000000` for the read data and the mask `0x00000000` for the read ECC word:

- `IMA_WRITE_DATA_0 = IMA_READ_DATA_0 ^ 0x81000000`, and
- `IMA_WRITE_DATA_1 = IMA_READ_DATA_1 ^ 0x00000000`.

As result, in this example test, `IMA_WRITE_DATA_0 = 0x81000004` and `IMA_WRITE_DATA_1 = 0x00000007`. The registers `IMA_WRITE_DATA_x` (with `x = 2,3,4`) are not used in this case.

User starts the write operation by setting the `IMA_ENABLE.EN` bit. This bit is a self-clear bit after two clocks. When the hardware clears this flag the write operation ends.

To allow accesses by other master (e.g. CPU), the user must de-select the memory location (`IMA_SLCT.ARRAY_SLCT = 0` `IMA_SLCT.ROW_SLCT = 0`). Afterward, a master, e.g. CPU, can read the selected memory location to read the corrupted data and detect the inserted double bit error.

3.2.7 Check relevant MEMU registers for DBE

In this example, the injected double bit error is correctly detected and reported to the MEMU. Indeed the valid bit is asserted (`PERIPH_RAM_UNCERR_STS0.VLD = 1`) indicating that the entry in the MEMU reporting table is valid. (`PERIPH_RAM_UNCERR_STS0 = 0xF7ED_400C` or `0xFBED_400C`). The assertion of the valid bit causes the uncorrectable error detect flag to be asserted (`ERR_FLAG.PR_UCE = 1`).

3.3 IMA testing SRAM (System RAM) ECC logic

According to the [Table 1](#) IMA is able to perform a 40-bits access on SRAM memory, where the first 32-bits are the data and the remaining 8-bits are the ECC of the address and data.

In case of read operations, the IMA module stores the read 40-bits in the registers IMA_READ_DATA_x (with x = 0,1):

- IMA_READ_DATA_0 contains data and
- IMA_READ_DATA_1 contains the relevant 8-bits ECC

The registers IMA_READ_DATA_x (with x = 2,3,4) are not used.

In a similar way, in case of write operation, IMA module stores the 40-bits to be written in the registers IMA_WRITE_DATA_x (with x = 0,1):

- IMA_READ_DATA_0 contains data word to be written, and
- IMA_READ_DATA_1 contains the relevant 8-bits ECC to be written.

As result, IMA module is able to program invalid data or ECC bits. Using this feature, the user can verify the integrity of the ECC logic of the selected peripheral.

SPC582B60 device has a 96KB SRAM. Moreover, the available SRAM is split in four blocks. The example focuses on the 4th RAM block whose addresses go from 0x400B_8000 to 0x400B_FFFF. This block is the SRAM_2_2 listed in [Table 1](#).

To verify the integrity of the ECC, the user can follow the following steps:

1. The SRAM is initialized with known data
2. Read operation by IMA is performed on the selected memory location
3. Write operation by IMA is performed on the selected memory location injecting an error
4. Read access by a master is performed on the selected memory location
5. Check relevant MEMU registers to verify the corrected detection of the error

3.3.1 SRAM initialization

In this example, the SRAM is initialized with data 0x0000xxxx (with xxxx = 0x0001,...,0x2000) starting from the address 0x400B_8000 to the address 0x400B_FFFC.

3.3.2 Read operation by IMA on the selected memory

This example considers the memory location at the address 0x400B_8008 as target.

After unlocking the IMA read access, the user selects the memory location:

1. IMA_SLCT.ARRAY_SLCT = 15 to selecte the 4th SRAM block (SRAM_2_2),
2. IMA_SLCT.ROW_SLCT = 2 to select the 3rd slot starting at address 0x400B808 and
3. IMA_CTRL.READ = 1

The user starts the reads operation by setting the IMA_ENABLE.EN bit (IMA_ENABLE.EN = 1). This bit is a self-clear bit after two clocks. The hardware ends the read operation when it clears this bit.

The read data are available in the registers IMA_READ_DATA_x (with x = 0,1). In this example test:

1. IMA_READ_DATA_0 = 0x03000000, and
2. IMA_READ_DATA_1 = 0x000000AF

where 0xAF is the 8-bits ECC calculated for these data.

Note that the computation of the 8-bits ECC includes the full target address 0x400B_8008 and that the data read by IMA are represented in big-endian format.

To allow access by other master (e.g. CPU), the user must de-select the memory location (IMA_SLCT.ARRAY_SLCT = 0 and IMA_SLCT.ROW_SLCT = 0).

3.3.3 Write operation by IMA on the selected memory inserting a SBE

To inject a single bit error in the memory location at the address 0x400B8008, after unlocking of the IMA write access, the user selects the memory location to be tested:

1. IMA_SLCT.ARRAY_SLCT = 15 to select the 4th SRAM block SRAM_2_2
2. IMA_SLCT.ROW_SLCT = 2 to select the 3rd slot starting at address 0x400B_8008, and
3. IMA_CTRL.READ = 0.

In this example, a single bit error is inserted in the 7th position of the data using the mask 0x40000000 for the read data with big-endian format and the mask 0x00000000 for the read ECC word by writing the registers.

1. IMA_WRITE_DATA_0 = IMA_READ_DATA_0^0x40000000, and
2. IMA_WRITE_DATA_1 = IMA_READ_DATA_1^0x00000000.

As result, in this example, IMA_WRITE_DATA_0 = 0x43000000 and IMA_WRITE_DATA_1 = 0x000000AF. The registers IMA_WRITE_DATA_x (with x = 2,3,4) are not used in this case.

User starts the write operation by setting the IMA_ENABLE.EN bit.

To allow access to by other master (e.g. CPU), the user must de-select the memory location (IMA_SLCT.ARRAY_SLCT = 0 and IMA_SLCT.ROW_SLCT = 0).

3.3.4 Read access by a master (e.g. CPU)

After the injecting of the error, the user perform a read access to the selected memory location to read the corrupted data, and trigger the hardware reaction.

3.3.5 Check relevant MEMU registers for SBE

In this example, the inserted single bit error is detected and reported to the MEMU. Indeed, the valid bit SYS_RAM_CERR_STS0.VLD = 1 is asserted indicating that the entry in MEMU reporting table is valid (SYS_RAM_CERR_ADDR0 = 0x400B_8008). The assertion of the valid bit causes the correctable error detect flag to be asserted (ERR_FLAG.SR_CE = 1).

Note that for SRAM the syndrome (SYS_RAM_CERR_STS0.BAD_BIT = 0x3C) is not reported to the MEMU and, as result, it cannot be used as indication of the bit position of the detected error. For further details refers to the defect DAN-0045130 included in the SPC582B60 Errata Sheet (3).

In case of single bit error in a certain memory location, the error is detected and the data is corrected by ECC logic before it is provided to the requesting master. As result, the memory location still holds corrupted data, as can be verified performing a read operation by IMA on the memory location.

3.3.6 Write operation by IMA on the selected memory inserting a DBE

To inject a double bit error in the memory location at the address 0x400B_8008, after the unlocking of the IMA write access), the user selects the memory location to be tested by the following fields:

1. IMA_SLCT.ARRAY_SLCT = 15 to select the 4th SRAM block SRAM_2_2
2. IMA_SLCT.ROW_SLCT = 2 to select the 3rd slot starting at address 0x400B_8008, and
3. IMA_CTRL.READ = 0.

In this example test, a double bit error is injected by using the mask 0x10000010 for the read data and the mask 0x00000000 for the read ECC word:

1. IMA_WRITE_DATA_0 = IMA_READ_DATA_0^0x10000010, and
2. IMA_WRITE_DATA_1 = IMA_READ_DATA_1^0x00000000.

As result, in this example test, IMA_WRITE_DATA_0 = 0x13000010 and IMA_WRITE_DATA_1 = 0x000000AF. The registers IMA_WRITE_DATA_x (with x = 2,3,4) are not used in this case.

User starts the write operation by setting the IMA_ENABLE.EN bit.

To allow access by other master (e.g. CPU), the users must de-select the memory location (IMA_SLCT.ARRAY_SLCT = 0 IMA_SLCT.ROW_SLCT = 0). Afterward, they perform a normal read access to the selected memory location to read the corrupted data and trigger the detection of the inserted double bit error.

3.3.7 Check relevant MEMU registers for DBE

In this example test, the inserted double bit error is correctly detected and reported to the MEMU. Indeed, the valid bit SYS_RAM_UNCERR_STS.VLD = 1 is asserted indicating that the entry in MEMU reporting table is valid (SYS_RAM_UNCERR_ADDR = 0x400B_8008). The assertion of the valid bit causes the uncorrectable error detect flag to be asserted (ERR_FLAG.SR_UCE = 1).

Note that the hardware triggers a machine check exception in case of double bit error in the SRAM detected while the CPU read a faulty location.

4 Conclusion

The capability of the IMA allows to access any RAM memory for the purpose of reading or modifying data, or both, including ECC check bits and this can be used to implement a device self-test verifying the ECC functionality, i.e. the correct working of the correction/detection mechanism, and also the correct working of the reporting/reaction mechanism.

5 Reference documents

1. STMicroelectronics. SPC582Bx 32-bit Power Architecture microcontroller for automotive vehicle body and gateway applications. (RM0403, DocID027949)
2. SPC58XXADPT64S Data brief. [DB3448, DocID031255].
3. SPC582Bx Errata Sheet. [ES0413, DocID031246].

6 Terms and Abbreviations

This document uses the following terms and abbreviations.

AIPS Peripheral

CAN Controller area network

CPU Central processing unit

DBE Double Bit Error

DMA Direct memory access

ECC Error correction code

E2E ECC End-To-End ECC

FCCU Fault collection and control unit

IMA Indirect memory access

MEMU Memory error management unit

RAM Random access memory

SBE Single Bit Error

SOC System-on-chip

SRAM System ram

7 Revision history

Table 3. Document revision history

Date	Revision	Changes
04-Apr-2018	1	Initial release.

IMPORTANT NOTICE – PLEASE READ CAREFULLY

STMicroelectronics NV and its subsidiaries ("ST") reserve the right to make changes, corrections, enhancements, modifications, and improvements to ST products and/or to this document at any time without notice. Purchasers should obtain the latest relevant information on ST products before placing orders. ST products are sold pursuant to ST's terms and conditions of sale in place at the time of order acknowledgement.

Purchasers are solely responsible for the choice, selection, and use of ST products and ST assumes no liability for application assistance or the design of Purchasers' products.

No license, express or implied, to any intellectual property right is granted by ST herein.

Resale of ST products with provisions different from the information set forth herein shall void any warranty granted by ST for such product.

ST and the ST logo are trademarks of ST. All other product or service names are the property of their respective owners.

Information in this document supersedes and replaces information previously supplied in any prior versions of this document.

© 2018 STMicroelectronics – All rights reserved

