AN5254
Application note

SPC58EHx/SPC58NHx FCCU fault sources and reaction

Introduction

This application note describes the FCCU input fault sources. Furthermore, for each of them, it describes how to verify the integrity of the error reaction path and the recommended methods to inject a fault.

The device mentioned in this document is the SPC58EHx/SPC58NHx (40 nm – ASIL D). Most of the concepts, however, are also valid for the other devices belonging to the 40nm and 55 nm families of SPC5 32-bit Automotive MCUs.

Before reading this document, the reader should have a clear understanding of the usage of FCCU. Refer to “Fault Collection and Control Unit (FCCU)” chapter in the SPC58EHx/SPC58NHx Microcontroller Reference Manual for further details on this module. Refer also to the SPC58EHx/SPC58NHx errata sheet.

Contents

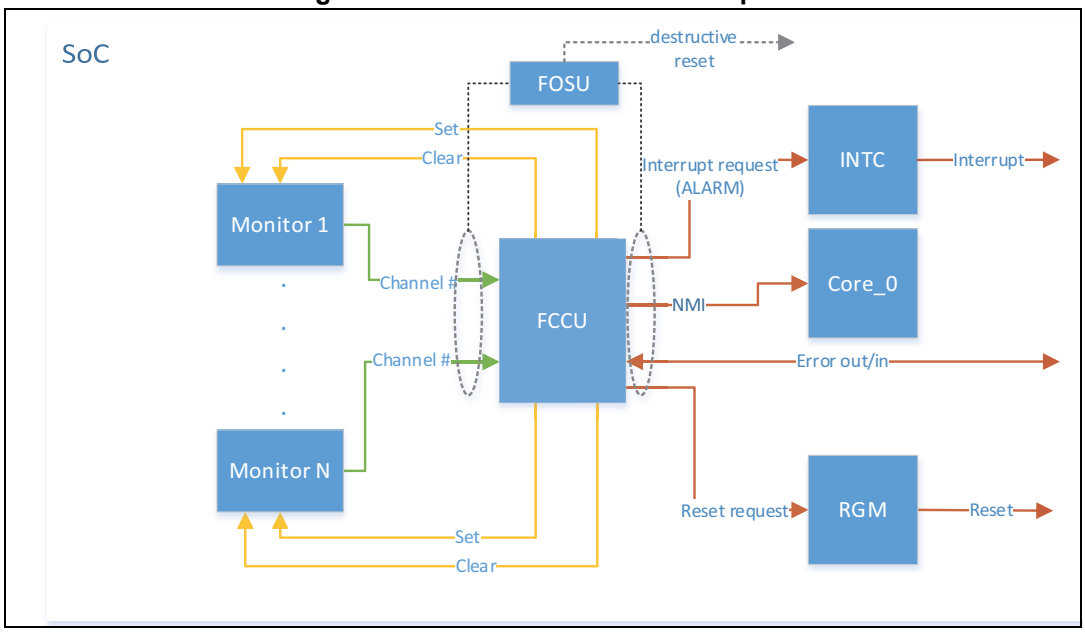
List of tables

List of figures

1 Overview

The FCCU is a crucial element of the functional safety concept of the SPC58 and SPC57 families of SPC5 32-bit Automotive MCUs. It is responsible for collecting and reacting to failure notifications coming from different modules indicated as monitors. Examples of monitors are CMU, MEMU, XBIC and so forth.

Figure 1. FCCU Monitor to reaction path



Note: Some monitors might miss the Set and Clear signals.

Figure 2 shows how the FCCU is connected to the other blocks of the SoC. The reader shall consider this figure – and all other figures in this document – as a block diagram that doesn't reflects exactly the physical implementation in the silicon.

In case of a fault, the FCCU can move the device into the safe state[(a)] without any CPU intervention. Since the FCCU and the whole error reaction path are prone to latent failures, the safety concept requires the execution of a software test to verify the integrity of the error reaction path. The user must run this software test for each used and testable channel of the FCCU at least once per Trip time[(b)].

This document goes through the list of the faults reported by the FCCU. For each of them, it describes how to test the reaction path to fulfill the requirement of the safety analysis. Note that the user can't test the error reaction path for few monitors.

Table 1 lists and describes all FCCU input fault sources for SPC58EHx/SPC58NHx.

a. It's worth noticing that our safety concept doesn't consider the Safe Mode of the Mode Entry as a safety states. The safety manual lists the considered safe states.

b. The FMEDA reports the Trip time considered by the Safety Analysis.

Table 1. List of faults

Channel #	Source	Description
0	PMC DIG	Temperature out of range from TSENS detector
1	PMC DIG	Supply voltage out of range from LVDs
2	PMC DIG	Supply voltage out of range from HVDs
3	PMC DIG	Digital PMC initialization error during DCF data load.
4	PMC DIG	Voltage detector BIST error
5	SSCM/FLASH	Transfer error (during the SSCM and STCU DCF loading) OR FLASH memory initialization error
6	STCU	BIST result - wrong signature (STCU Unrecoverable Fault)
7	STCU	BIST result - wrong signature (STCU Recoverable Fault)
8	STCU	Unwanted activation of the online self-test
9	GLUE LOGIC	JTAG or NPC not in reset or activation of dangerous debug functionality. Spurious activation of SSCM
10	IAHBG	Error in the frequency signaled by the gasket between the XBAR 2 and the XBAR 0
11	CORE 2	Lockstep out of sync due to a fault in the Core 2 or associated DSMC
12	INTC	Lockstep out of sync due to a fault in the INTC
13	IAHBG	Error in the frequency signaled by the gasket between the XBAR 2 and PRAMC0 Port 1
14	IAHBG	Error in the frequency signaled by the gasket between the XBAR 2 and PRAMC1 Port 1
15	SWT	Software watchdog timer 3 reset request
16	SWT	Software watchdog timer 2 reset request
17	SWT	Software watchdog timer 1 reset request
18	SWT	Software watchdog timer 0 reset request
19	MEMU	A new error is recorded in the MEMU SysRAM correctable error table
20	MEMU	A new error is recorded in the MEMU SysRAM uncorrectable error table
21	MEMU	A new error is recorded in the MEMU SysRAM overflow error table
22	MEMU	A new error is recorded in the MEMU Peripheral RAM correctable error table
23	MEMU	A new error is recorded in the MEMU Peripheral RAM uncorrectable error table
24	MEMU	A new error is recorded in the MEMU Peripheral RAM overflow error table

Table 1. List of faults (continued)

Channel #	Source	Description
25	MEMU	A new error is recorded in the MEMU FLASH correctable error table
26	MEMU	A new error is recorded in the MEMU FLASH uncorrectable error table
27	MEMU	A new error is recorded in the MEMU FLASH overflow error table
28	IMA	IMA is active
29	SMPU	SMPU XBAR 0 Monitor incorrectly refuses an access
30	SMPU	SMPU XBAR 1 Monitor incorrectly refuses an access
31	SMPU	SMPU XBAR 0 Monitor correctly refuses an access
32	SMPU	SMPU XBAR 1 Monitor correctly refuses an access
33	CORE 0	I-MEM address feedback error
34	CORE 0	D-MEM address feedback error
35	CORE 0	I-CACHE address feedback error
36	CORE 0	D-CACHE address feedback error
37	CORE 1	I-MEM address feedback error
38	CORE 1	D-MEM address feedback error
39	CORE 1	I-CACHE address feedback error
40	CORE 1	D-CACHE address feedback error
41	CORE 2	I-MEM address feedback error
42	CORE 2	D-MEM address feedback error
43	CORE 2	I-CACHE address feedback error
44	CORE 2	D-CACHE address feedback error
45	DMA 0	DMA 0 TCD RAM address feedback error
46	DMA 1	DMA 1 TCD RAM address feedback error
47	PLL DIG	PLL0 Loss of lock error
48	PLL DIG	PLL1 Loss of lock error
49	CMU	Loss of XOSC clock (OLR signal from CMU0)
50	CMU	System clock frequency out of range (FHH or FLL signal from CMU0)
51	CMU	Platform clock frequency out of range (FHH or FLL signal from a CMU belonging to CMU_Platform group)
52	CMU	Other clocks frequency out of range (FHH or FLL signal from a CMU belonging to CMU_other group)
53	ETHERNET 1	EDC after ECC error
54	XBAR 0	Crossbar integrity error signaled by XBIC
55	XBAR 1	Crossbar integrity error signaled by XBIC

Table 1. List of faults (continued)

Channel #	Source	Description
56	FLASH 0	EDC after ECC error from FLASH array
57	PFLASHC 0	EDC after ECC error from FLASH controller
58	FLASH 0	FLASH Address Encoding Error
59	PFLASHC 0	FLASH Controller Address feedback checker error
60	FLEXRAY 0	EDC after ECC error
61	FLASH 0 / 1	FLASH read reference error
62	FLASH 1	EDC after ECC error from FLASH array
63	PFLASHC 1	EDC after ECC error from FLASH controller
64	FLASH 1	FLASH Address Encoding Error
65	PFLASHC 1	FLASH Controller Address feedback checker error
66	XBAR 2	Crossbar integrity error signaled by XBIC
67	PFLASHC 1	OTA functionality on PFLASHC
68	PRAM 0	System RAM Address Feedback or RAM Late-Write Buffer mismatch or EDC after ECC error for RMW operations
69	PFLASHC 0	OTA functionality on PFLASHC
70	PRAM 1	System RAM Address Feedback or RAM Late-Write Buffer mismatch or EDC after ECC error for RMW operations
71	CORE 0	Indication of an hardware fault affecting the control signals of the DSMC associated to the core
72	PRAM 2	System RAM Address Feedback or RAM Late-Write Buffer mismatch or EDC after ECC error for RMW operations
73	CORE 1	Indication of a hardware fault affecting the control signals of the DSMC associated to the core
74	PRAM 3	System RAM Address Feedback or RAM Late-Write Buffer mismatch or EDC after ECC error for RMW operations
75	IAHBG	Error in the frequency signaled by the gasket between the DMSC for the core 0 and the XBAR 1
76	TCU	Test circuitry unit activation
77	IAHBG	Error in the frequency signaled by the gasket between the instruction bus of the core 0 and the XBAR 1
78	IAHBG	Error in the frequency signaled by the gasket between the DMSC for the core 1 and the XBAR 0
79	IAHBG	Error in the frequency signaled by the gasket between the instruction bus of the core 1 and the XBAR 0
80	CORE 0	Machine check exception indication
81	CORE 1	Machine check exception indication
82	CORE 2	Machine check exception indication
83	DMA 0	EDC after ECC error

Table 1. List of faults (continued)

Channel #	Source	Description
84	IAHBG	Error in the frequency signaled by the gasket between the XBAR 0 and the AIPS 0
85	AIPS 0	EDC after ECC error on a transaction through the AIPS
86	IAHBG	Error in the frequency signaled by the gasket between the XBAR 1 and the AIPS 1
87	AIPS 1	EDC after ECC error on a transaction through the AIPS
88	IAHBG and AIC	Error in the frequency signaled by the gasket between the XBAR 1 and the AIPS 2 or integrity error reported by AIC
89	AIPS 2	EDC after ECC error on a transaction through the AIPS
90	CONCENTRATOR 0	Internal Crossbar integrity error signaled by its internal XBIC
91	DMA 1	EDC after ECC error
92	MC_RGM	Safe mode entry request from RGM
93	COMPENSATION CELLS	Pad Compensation Disabled
94	GLUE LOGIC	Error input pin (from the external world)
95	CORE 1	MasterID monitor for Core Data bus
96	CORE 1	MasterID monitor for Core Instruction bus
97	ETHERNET 0	EDC after ECC error
98	SMPU	SMPU Backdoor XBAR Monitor correctly refuses an access
99	SMPU	SMPU Backdoor XBAR Monitor incorrectly refuses an access
100	SWT	First Timeout event of SWT3 which is also mapped to INTC
101	SWT	First Timeout event of SWT2 which is also mapped to INTC
102	SWT	First Timeout event of SWT1 which is also mapped to INTC
103	SWT	First Timeout event of SWT0 which is also mapped to INTC
104(1)	USB	ECC errors on Master or EDC after ECC for Master / Slave
105	Octal SPI	EDC after ECC error
106	eMMC	EDC after ECC error
107	SMPU	SMPU XBAR 2 Monitor correctly refuses an access
108	SMPU	SMPU XBAR 2 Monitor incorrectly refuses an access
109	SSCM	Error while doing OTA signature scanning
110	HSM	Concentrator HSM EDC error
111	CORE 2	Indication of disablement of lockstep mode
112	INTC	Indication of disablement of lockstep mode
113(1)	ZIPWIRE 0	EDC after ECC error
114(1)	ZIPWIRE 1	EDC after ECC error

Table 1. List of faults (continued)

Channel #	Source	Description
115	FLASH 0 / 1	FLASH internal unrecoverable error
116	CORE 0	MasterID monitor for Core Data bus
117	CORE 0	MasterID monitor for Core Instruction bus
118	IAHBG	Error in the frequency signaled by the gasket between the Core 0 backdoor and the Backdoor XBAR
119	IAHBG	Error in the frequency signaled by the gasket between the Core 2 backdoor and the Backdoor XBAR
120	IAHBG	Error in the frequency signaled by the gasket between the Concentrator 0 and the XBAR 0
121	IAHBG	Error in the frequency signaled by the gasket between the Concentrator 1 and the XBAR 1
122	IAHBG	Error in the frequency signaled by the gasket between the PRAMC 0 P0 and the XBAR 0
123	IAHBG	Error in the frequency signaled by the gasket between the PRAMC 1 P0 and the XBAR 0
124	IAHBG	Error in the frequency signaled by the gasket between the PRAMC 2 P0 and the XBAR 1
125	IAHBG	Error in the frequency signaled by the gasket between the PRAMC 3 P0 and the XBAR 1
126	IAHBG	Error in the frequency signaled by the gasket between the Core 1 backdoor and the XBAR 0
127	IAHBG	Error in the frequency signaled by the gasket between the XBAR 2 and the XBAR 0

1. Only for cut 1. In the cut 2 of the device this functionality is not available and the channel is not connected.

Before the safety application starts, the user shall configure and set a proper reaction for each FCCU input fault source. See “FCCU register reset values” paragraph in SPC58EHx/SPC58NHx Microcontroller Reference Manual for the device default configuration.

Possible FCCU’s reactions to failure are:

- Internal reactions[(c)]:
  - No reset reaction
  - IRQ
  - Long or Short functional reset
  - NMI to Core 0
- External reaction:
  - Error out (EOUT) signaling

c. The user can configure separately the internal reaction for each FCCU input.

Once it is enabled[d], the FCCU controls EOUT pins that signs the status of the MCU without any intervention of the core.

A dedicated module - i.e., the FOSU - monitors the integrity of the FCCU. The FOSU triggers a destructive reset if its internal counter reaches a timeout before the FCCU takes a reaction to an incoming – and enabled[e] – fault. The FOSU does not require any configuration done by the user. A functional reset has no impact on the FCCU configuration.

d. The EOUT pins are enabled by the FCCU_SET_AFTER_RESET flag of the FCCU_CFG register.

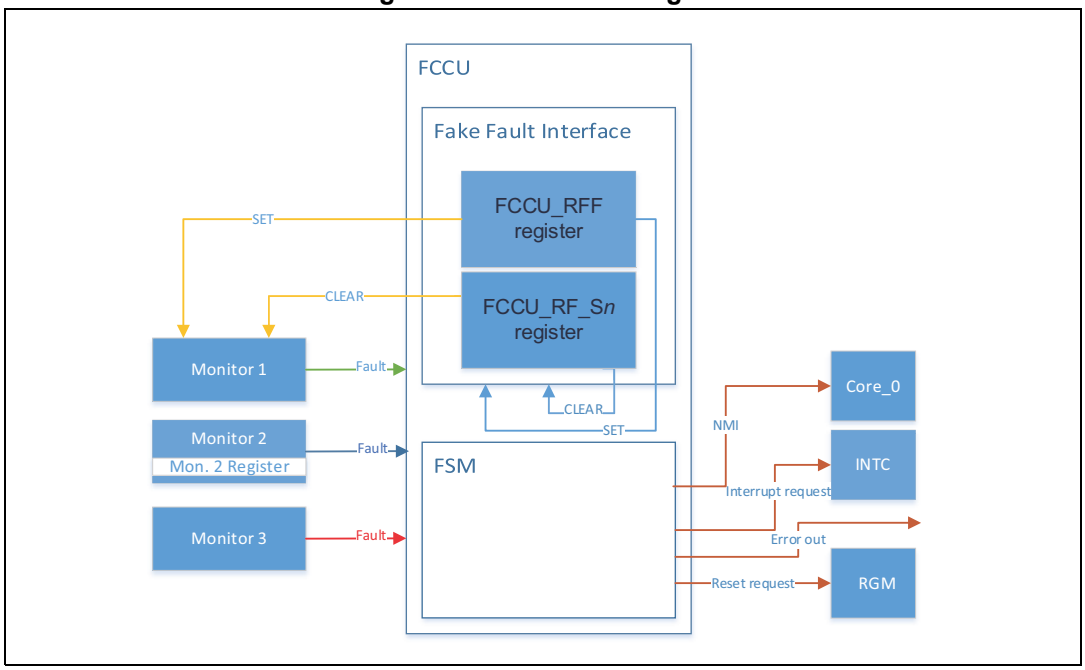
e. The FOSU ignores the incoming faults that aren't enabled through the corresponding bit in the FCCU_RF_En register.

2 FCCU Fault injection, clearing and fake fault interface

The application can use the fault injection to diagnose physical defects affecting the connections between the hardware monitors of the used module and the FCCU. The procedure to inject a fault depends on the specific monitor. We can distinguish among three different sets of fault inputs:

1. Faults that are injectable by using the FCCU fake fault interface (e.g., “Monitor 1” in *Figure 2*)
2. Faults that are injectable by using a SW procedure to stimulate the fault (e.g., “Monitor 2” in *Figure 2*)
3. Faults that are not injectable (e.g., “Monitor 3” in *Figure 2*)

Figure 2. FCCU Inner diagram



For the first group, the FCCU can trigger a fault event directly in the monitor (even if no real failure occurs) through the fake fault interface.

To generate this fault event, an additional - and optional[f] - signal is available (SET signal in *Figure 2*, yellow arrow). The user can inject a fake fault by asserting the bit corresponding to the channel under test into the FCCU_RFF register. To clear a fake fault directly in the monitor, an additional - and optional[f] - signal is available (CLEAR signal in *Figure 2*, yellow arrow). The de-assertion of the FCCU_RF_Sn status bit indicates that the software has appropriately cleared the fake fault in the monitor.

Some monitors miss the SET and CLEAR signals. In this case, the write operation into the FCCU_RFF register doesn’t affect the monitor but only the FCCU reaction[g]. With such a

f. Not all monitors embed this interface. When available, fake fault injection method is suggested in the following sections.

g. SET/CLEAR signals in *Figure 2: FCCU Inner diagram*, blue arrow.

method, the user can debug the reaction of the FCCU, but they can't verify the integrity of the connection between these monitors and the FCCU.

For the second group, the application can trigger a fault event writing into a specific register of the monitor or using an SW procedure (e.g., injection of ECC errors, setting wrong thresholds). To clear a fault that belongs to this group, the application must clear the fault indication in the monitor[h] and afterward de-assert the corresponding FCCU_RF_Sn status bit.

It is not possible to test the connection for the faults that belong to the third group. The safety analysis considers and assesses this limitation.

The reaction path test procedure must also include the check of FCCU's reaction (EOUT, Interrupt, NMI, reset request), at least for one FCCU input fault.

Depending on the monitor, the fault indication is a pulse (edge triggered fault) or a constant value (level triggered fault).

The fault management shall consider that the user can configure a fault as:

- HW recoverable fault. I.e., the fault status within the FCCU remains asserted until the monitor keeps the fault indication asserted. As soon as the monitor clears the fault indication, it also clears the fault status within the FCCU.
- SW recoverable fault. I.e., the fault status within the FCCU remains asserted until the software clears it even if the monitor de-asserts the fault indication.

The generic recommendation is to configure all faults as SW recoverable. In such a way, the FCCU clears the respective status flag only after an explicit request from the software. In the case of HW recoverable setting, the status flag automatically clears, and the application may not react appropriately to the incoming fault.

h. This is not necessary for monitors that have a pulse fault indication

3 Faults description

The following sections describe all the faults collected by the FCCU for SPC58EHx/SPC58NHx device and how - if possible - to inject and clear them.
The coming figures adopt the following convention:

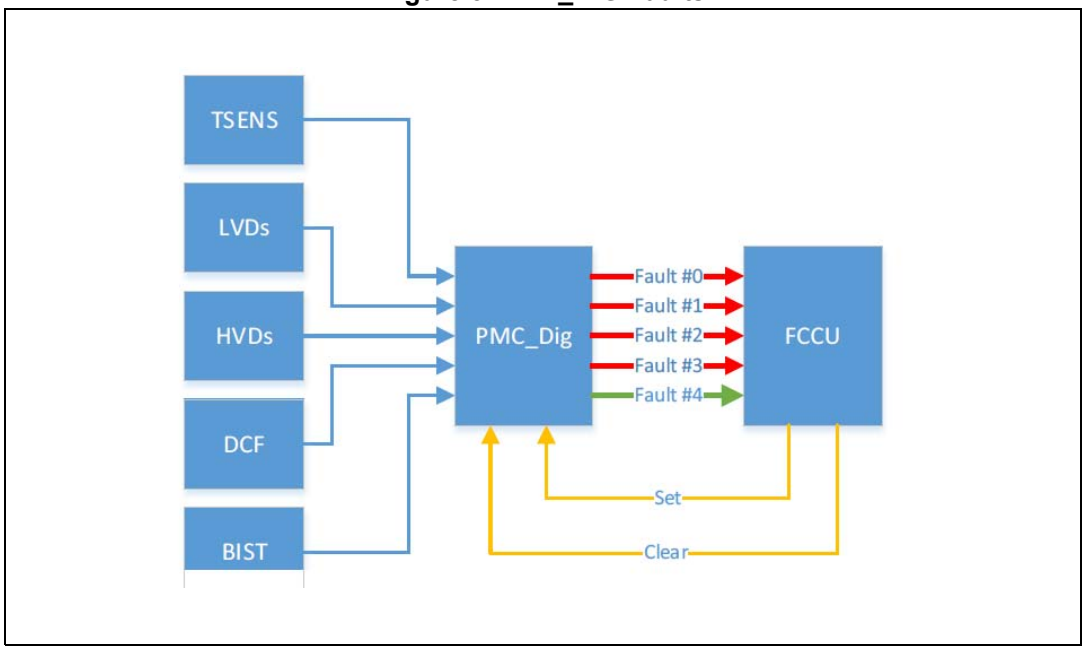
- A GREEN arrow marks the faults as injectable by the FCCU fake fault interface
- A BLUE arrow marks the faults as injectable by using a SW procedure to stimulate the fault
- A RED arrow marks the faults that the user can't inject

Refer to device SPC58EHx/SPC58NHx Microcontroller Reference Manual for further details on the mentioned modules.

3.1 PMC_DIG Faults

PMC_DIG is the source of five different FCCU input faults.

Figure 3. PMC_DIG Faults



3.1.1 Temperature out of range from TSENS (Fault #0)

The temperature sensor generates an output voltage that is proportional to the internal junction temperature of the device. When this temperature exceeds the defined thresholds[(i)], the PMC_DIG forwards this fault to the FCCU.

i. There are three fixed thresholds: TS0, TS1 and TS2. User can configure the reaction to them in the PMC_DIG FEE_TD register.

The SSCM loads the trim values of the temperature sensor from the FLASH by the DCF record interface during the boot phase.

The user can't inject this fault.

3.1.2 Voltage out of range from LVDs (Fault #1)

The PMC_DIG forwards this fault to the FCCU when an LVD detects a supply voltage that drops below the defined threshold[j].

The MCU embeds several LVDs (see SPC58EHx/SPC58NHx Microcontroller Reference Manual for further details on LVDs). Their output signals are OR'ed before arriving to the FCCU.

The SSCM loads their trim values from the FLASH by the DCF record interface during the boot phase.

The user can't inject this fault.

3.1.3 Voltage out of range from HVDs (Fault #2)

The PMC_DIG forwards this fault to the FCCU when an HVD detects a supply voltage that raises above the defined threshold[k]. The MCU embeds several HVDs (see SPC58EHx/SPC58NHx Microcontroller Reference Manual for further details on HVDs). Their output signals are OR'ed before arriving to the FCCU.

The SSCM loads the trim values from the FLASH by the DCF record interface during the boot phase.

The user can't inject this fault.

3.1.4 PMC_Dig DCF Safety Error (Fault #3)

DCF records are used to configure specific registers in the device during system boot. For example, trimming values used by analog modules (e.g. PMC, Temperature Sensor and ADC Bandgap). If an error occurs while the SSCM is loading the DCF records values into the PMC, the PMC_DIG forwards this fault to the FCCU.

The user can't inject this fault.

3.1.5 Voltage detector BIST (Fault #4)

This BIST is a testing procedure initiated by software. The voltage detector BIST verifies the integrity of all the supply voltage monitors. If the BIST fails, the PMC_DIG forwards this fault to the FCCU.

The user can inject this fault by the FCCU fake fault interface.

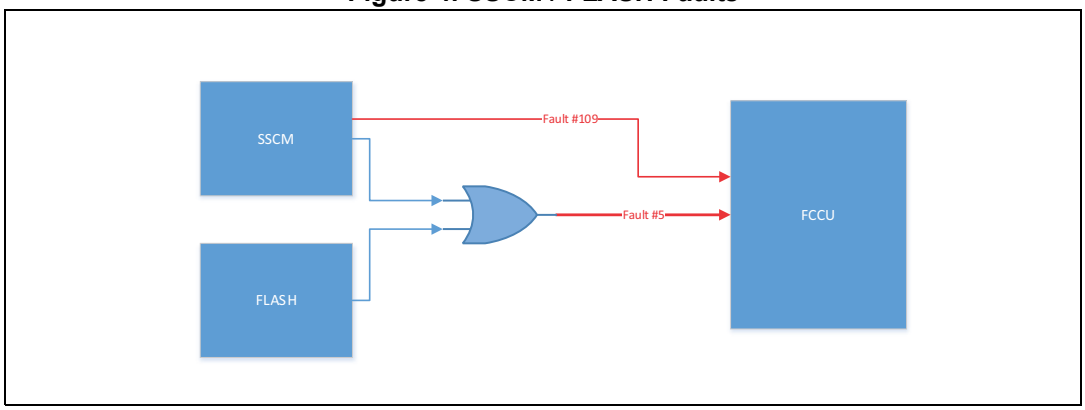
j. Corresponding bit in the PMG_Dig FEE_LVx must be enabled.

k. Corresponding bit in the PMG_Dig FEE_HVx must be enabled.

3.2 SSCM/FLASH Fault

The SSCM reads the system configuration from the FLASH[l] and writes it to the various clients inside the microcontroller.

Figure 4. SSCM / FLASH Faults



3.2.1 STCU Configuration OR FLASH memory initialization error (Fault #5)

A fault can occur while the SSCM loads the STCU configuration. In this case, the SSCM forwards this fault to the FCCU. Moreover, an unexpected condition - e.g., ECC double-bit detections on the reset reads can occur within the FLASH module during its initial configuration. In this case, the FLASH forwards this fault to the FCCU.

The FCCU Fault #5 is the resulting OR between the previous two signals.

The user can't inject this fault.

3.2.2 Error while doing OTA signature scanning (Fault #109)

A fault can occur while the SSCM loads the Flash block where is written the OTA signature indicating the active context[m]. In this case, the SSCM forwards this fault to the FCCU.

The user can't inject this fault.

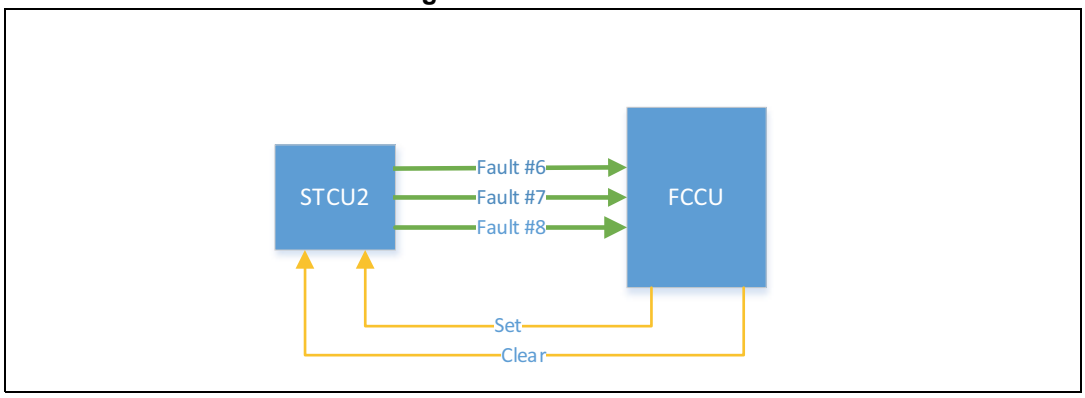
3.3 STCU Faults

The STCU is a programmable hardware module that controls the execution of the Self-Test during both the Off-line and/or On-line procedure. The STCU is the source of three different FCCU input faults.

l. System configuration is mainly saved as DCF records that are located either in the Test or UTest sectors of the flash.

m. User can refer to the "Over the air (OTA) support" section of the reference manual

Figure 5. STCU2 Faults



3.3.1 BIST result - wrong signature (STCU Unrecoverable Fault) (Fault #6)

3.3.2 BIST result - wrong signature (STCU Recoverable Fault) (Fault #7)

If the BIST detects a fault that the user configures as unrecoverable[(n)] or recoverable[(n)], the STCU forwards this fault to the FCCU. Although BIST targets permanent faults, a transient one may trigger a fault event.

Therefore, re-running the self-test may be the appropriate action to this fault. The STCU triggers this kind of fault only during BIST execution.

The user can inject this fault by the FCCU fake fault interface.

3.3.3 Unwanted activation of the BIST during the execution of the user application (Fault #8)

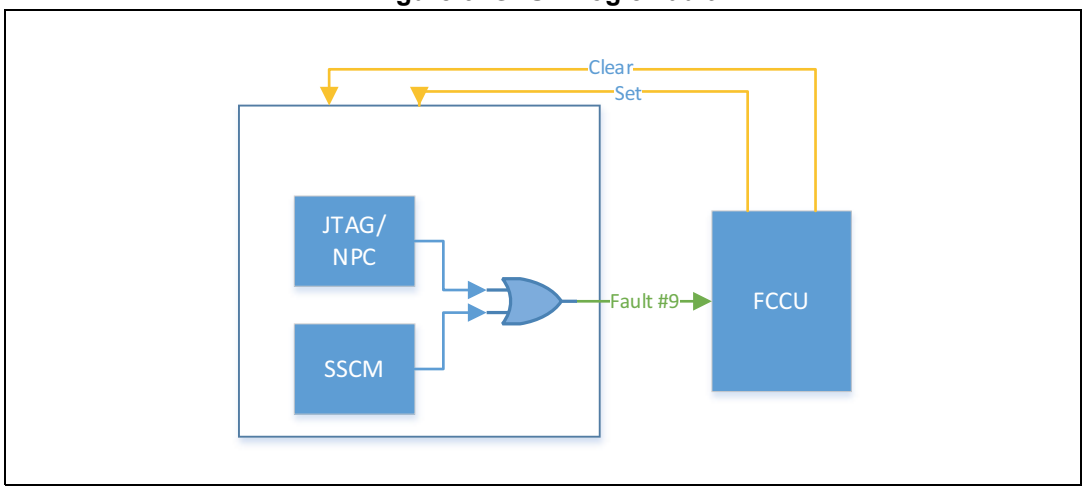
If the online self-test activates when the RUNSW field of the register STCU_RUNSW is set to ‘0’, The STCU detects this unexpected condition and forwards it to the FCCU.

The user can inject this fault by the FCCU fake fault interface.

n. The user shall configure the STCU to trigger either a recoverable or an unrecoverable fault if the BIST fails. This configuration is done by programming the proper DCF records.

3.4 GLUE LOGIC Faults

Figure 6. GLUE Logic Fault

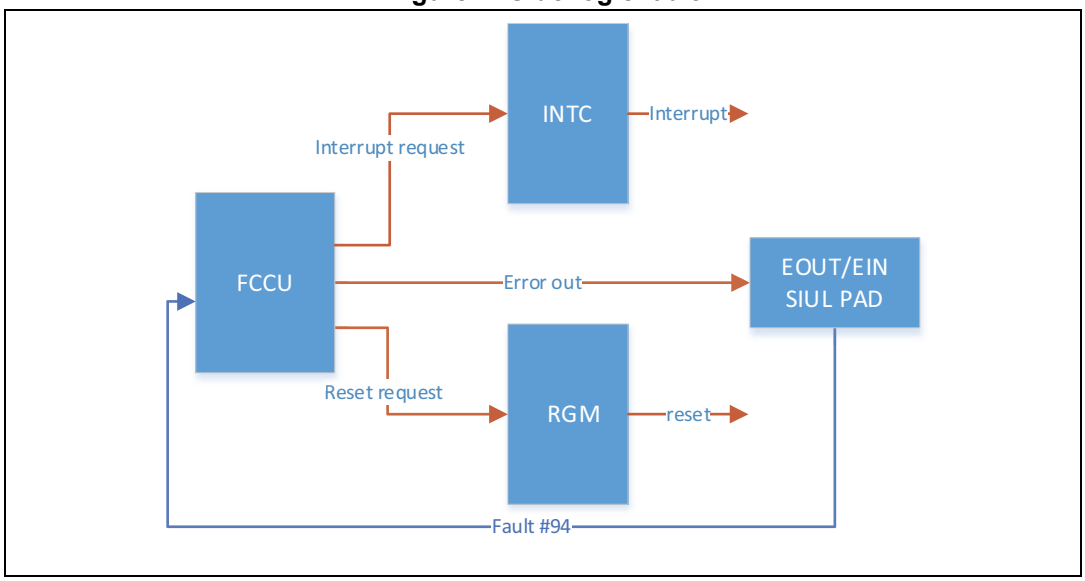


3.4.1 JTAG or NPC not in reset or activation of dangerous debug functionality. Spurious activation of SSCM (Fault #9)

The JTAG/NPC module monitors the unwanted activation of the JTAG/DEBUG Mode. If this event occurs, a dedicated logic forwards it to the FCCU. The hardware also monitors the unwanted activation of the SSCM and, if this event occurs, a dedicated logic forwards it to the FCCU. The FCCU Fault #9 is the resulting OR between the previous two signals.

The user can inject this fault by the FCCU fake fault interface.

Figure 7. Glue logic fault



3.4.2 External activation of EIN (Fault #94)

If an external device pulls down the EOUT/EIN pin, the MCU receives a notification of an external faulty condition. A dedicated logic forwards this fault to the FCCU.

The EOUT signal drives the EIN signal through an internal loopback. This connection results in a loop where if the FCCU is in the FAULT state, it drives the EOUT which in turn drives the EIN.

The user can inject this fault by a SW procedure that asserts the EOUT / EIN loopback, driving the EOUT through the FCCU_SET_CLEAR field of FCCU_CFG register.

To clear the fault, the SW procedure must break the EOUT/EIN loopback by setting the GPIO mode of the EOUT pin for a time longer than the EOUT time to expire (T_min set using the FCCU_XTMR register).

3.5 Intelligent AHB Gasket (IAHBG) Faults

The IAHBG modules do intelligent bridging between different clock domains.

3.5.1 Frequency error between XBAR 2 and XBAR 0 (Fault #10)

3.5.2 Frequency error between XBAR 2 and PRAMPC0 Port 1 (Fault #13)

3.5.3 Frequency error between XBAR 2 and PRAMPC1 Port 1 (Fault #14)

3.5.4 Frequency error between XBAR 1 and the Core 0 DSMC (Fault #75)

3.5.5 Frequency error between XBAR 1 and the Core 0 IBus (Fault #77)

3.5.6 Frequency error between XBAR 0 and the Core 1 DSMC (Fault #78)

3.5.7 Frequency error between XBAR 0 and the Core 1 IBus (Fault #79)

3.5.8 Frequency error between Backdoor XBAR and the Core 0 backdoor (Fault #118)

3.5.9 Frequency error between Backdoor XBAR and the Core 2 backdoor (Fault #119)

3.5.10 Frequency error between XBAR 0 and the Concentrator 0 (Fault #120)

3.5.11 Frequency error between XBAR 1 and the Concentrator 1 (Fault #121)

3.5.12 Frequency error between XBAR 0 and the PRAMC 0 P0 (Fault #122)

3.5.13 Frequency error between XBAR 0 and the PRAMC 1 P0 (Fault #123)

3.5.14 Frequency error between XBAR 1 and the PRAMC 2 P0 (Fault #124)

3.5.15 Frequency error between XBAR 1 and the PRAMC 3 P0 (Fault #125)

3.5.16 Frequency error between XBAR 0 and the Core 1 backdoor (Fault #126)

3.5.17 Frequency error between XBAR 0 and the XBAR 2 (Fault #127)

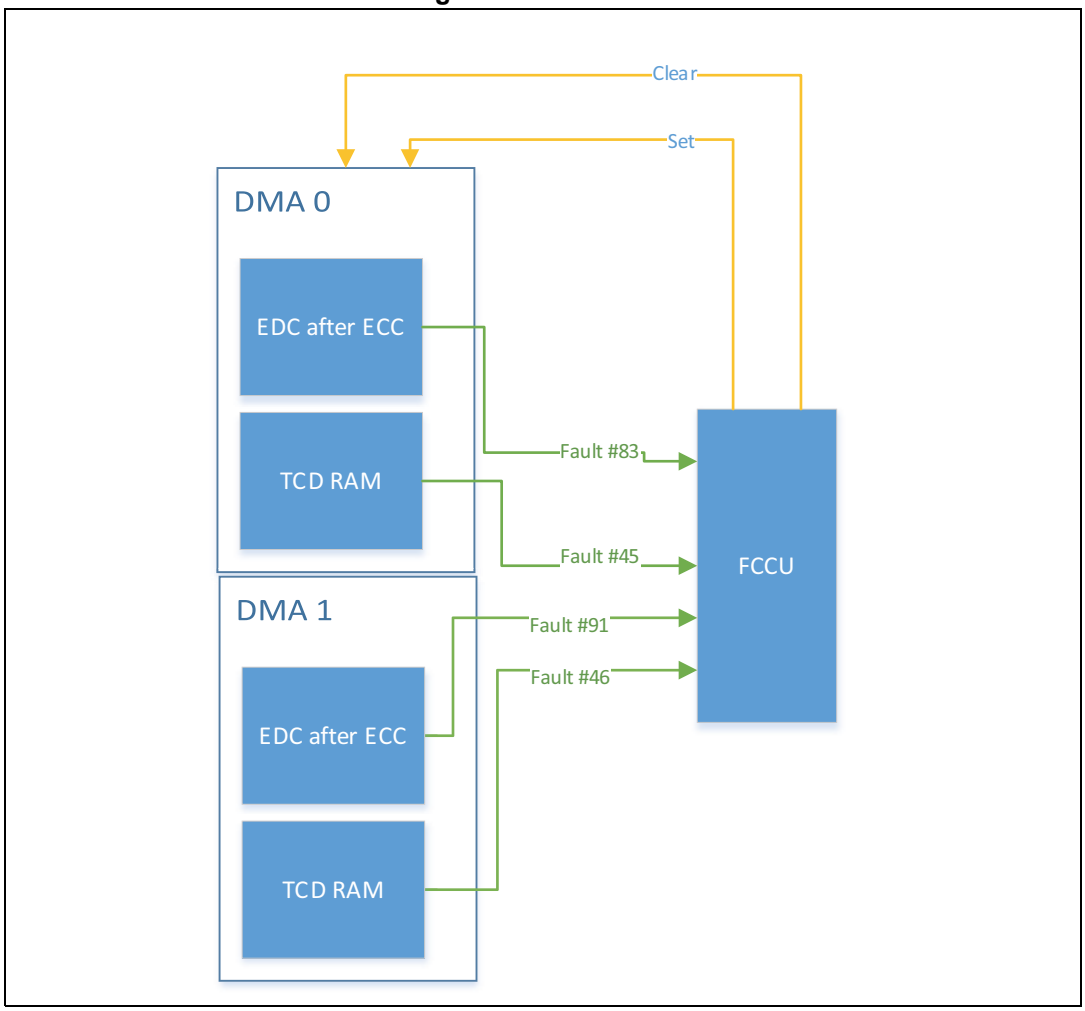
For all the frequency error faults above, the IAHBG signals to the FCCU whether a hardware fault results in a wrong frequency conversion between the different domains.

The user can’t inject these faults.

3.6 DMA Faults

The eDMA controller is a module capable of performing complex data transfers with minimal intervention of a host processor.

Figure 8. DMA Faults



3.6.1 DMA 0/1 TCD EDC after ECC (Fault #83 and Fault #91)

The EDC after ECC signals to the FCCU whether a hardware fault in the ECC logic of the DMA results in a corrupted ECC correction.

The user can inject this fault by the FCCU fake fault interface.

3.6.2 DMA 0/1 TCD Ram feedback checker (Fault #45 and Fault #46)

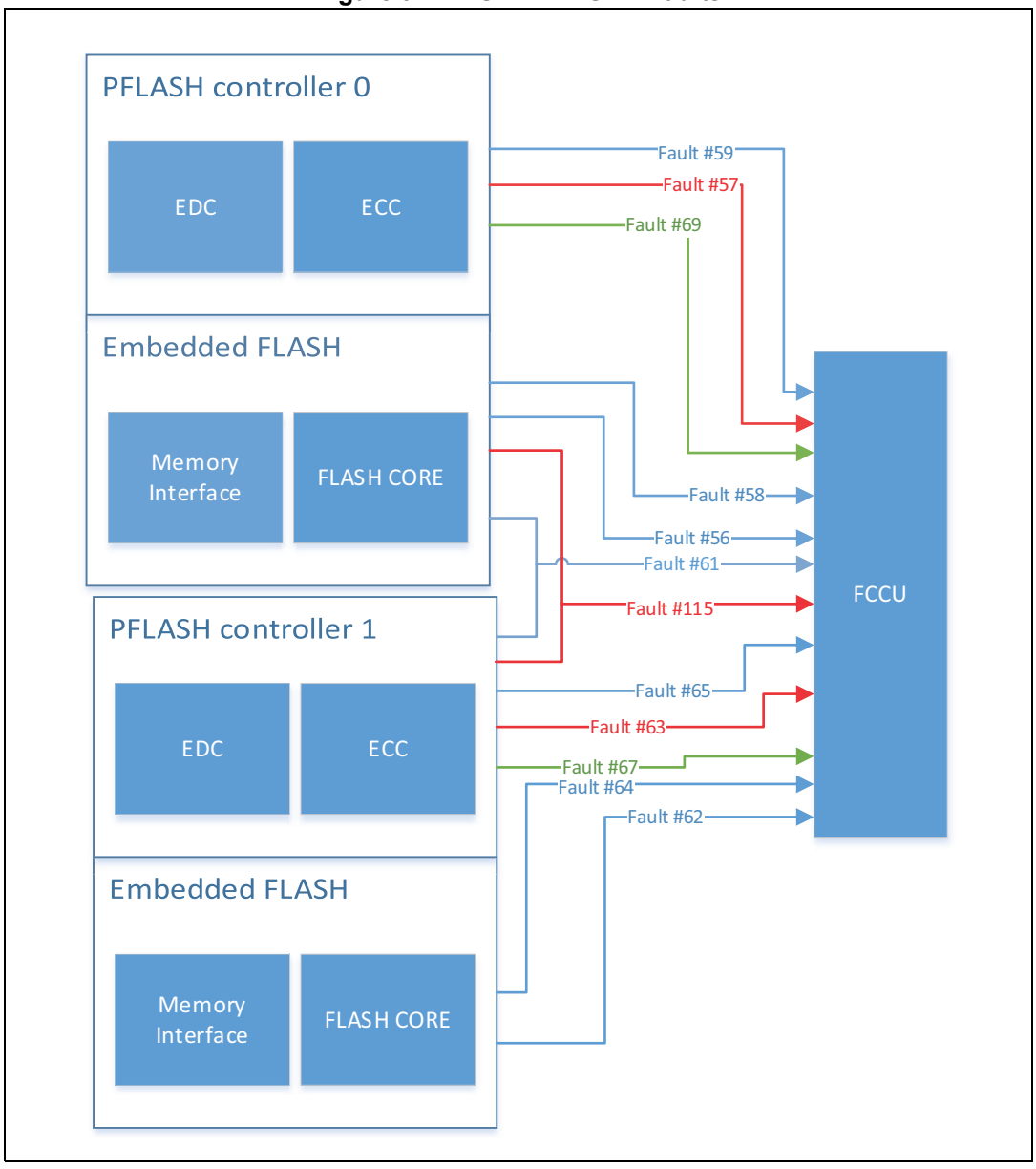
The feedback checker is a hardware that verifies the integrity of the signals that control the access the DMA TCD RAM. If latched control signals don't match with the ones initially sent to DMA TCD RAM, the feedback checker forwards this fault to the FCCU.

The user can inject this fault by the FCCU fake fault interface.

3.7 FLASH/PFLASHC Faults

The PFLASH controller is the interface between the FLASH memory and the crossbar switch. It provides FLASH configuration and control functions. The SPC58EHx/SPC58NHx embeds two PFLASHC controllers, each of them connected to a FLASH memory array.

Figure 9. FLASH/PFLASHC Faults



3.7.1 FLASH internal unrecoverable error (Fault #115)

The FLASH forwards this fault to the FCCU in case one of the following unrecoverable errors occurs:

- ECC errors on flash internal reads during configuration loading (startup);
- ECC errors on flash internal reads during firmware(o) copy (startup);

Double ECC errors on KRAM(p) during internal self-check routine (always running).

The user can't inject this fault.

3.7.2 FLASH read reference error (Fault #61)

The FLASH monitors its internal current and voltage references that are in common for both FLASH 0/1. In case one of these values is out of the allowed range, the FLASH forwards this event to FCCU.

The user can inject this fault by a SW procedure that accesses to the FLASH address in the Customer Programmable Detection area in the UTEST block.

3.7.3 EDC after ECC for FLASH Array (Fault #56 and Fault #62)

In case a hardware fault occurs in the ECC logic of the FLASH memory resulting in a corrupted ECC correction, the PFLASHC forwards this fault to FCCU. The user can inject this fault by a SW procedure that accesses to the FLASH address in the Customer Programmable Detection area in the UTEST block.

3.7.4 EDC after ECC for FLASH Controller (Fault #57 and Fault #63)

In case a hardware fault occurs in the ECC logic of the FLASH controller resulting in a corrupted ECC correction, the PFLASHC forwards this fault to FCCU.

The user can't inject this fault.

3.7.5 FLASH Encoding Error (Fault #58 and Fault #64)

In case a hardware fault occurs resulting in a corrupted FLASH memory access, the PFLASHC forwards this fault to FCCU.

The user can inject this fault by a SW procedure that accesses to the FLASH address in the Customer Programmable Detection area in the UTEST block.

3.7.6 FLASH Controller Address Feedback Checker Error (Fault #59 and Fault 65)

In case a hardware fault occurs resulting in a mismatch between the address from the crossbar and the feedback address from the FLASH, the PFLASHC forwards this fault to FCCU.

The user can inject this fault by a SW procedure that accesses to the FLASH address in the Customer Programmable Detection area in the UTEST block[q].

3.7.7 OTA functionality on PFLASHC 0/1 (Fault #67 and Fault #69)

When OTA is enabled and address translation is detected in the corresponding PFLASHC the hardware signals it to the FCCU. The user can inject this fault by the FCCU fake fault interface.

---

o. This is the PFLASH controller firmware and doesn't refer to the user firmware. It's not accessible to the user.

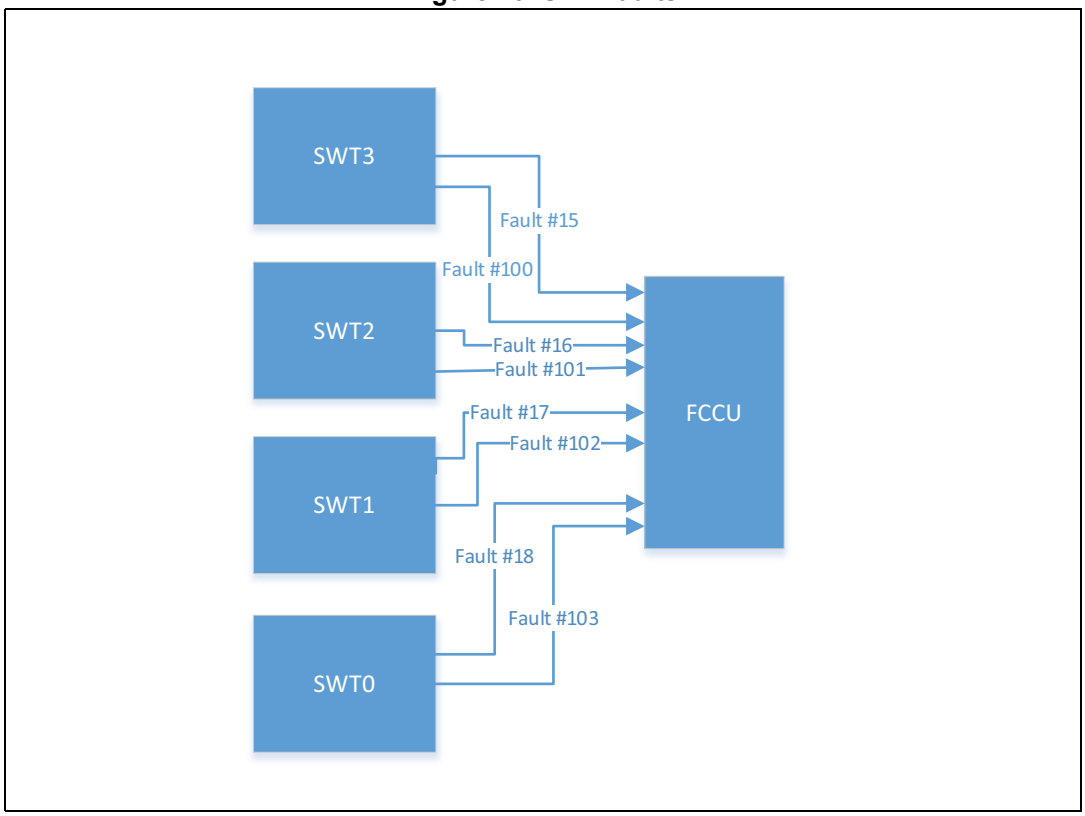
p. This is a RAM internal to the PFLASHC not visible to the user.

q. if an Address encoding error is injected (i.e., faults #58 or #64), the address feedback to PFLASHC is incorrect. Therefore, also PFLASHC triggers the faults to the FCCU (i.e., Fault #59 and Fault 65).

3.8 SWT Faults

The SWT is a module that can prevent system lockup in situations such as software getting trapped in an endless loop or a bus transaction which fails to terminate. When enabled, the SWT requires periodic execution of a watchdog servicing operation. The servicing operation resets the timer to a specified time-out period. If the software doesn't service the SWT before the timer expires, the SWT generates an interrupt or a reset according to its configuration.

Figure 10. SWT faults



3.8.1 SWT3 reset request (Fault #15)

3.8.2 SWT2 reset request (Fault #16)

3.8.3 SWT1 reset request (Fault #17)

3.8.4 SWT0 reset request (Fault #18)

If the SWT reaches a timeout[(r)] the HW forwards this fault to FCCU.

The user can inject this fault by a SW procedure that enables the SWT0 and doesn't service it. Note that the user can clear this fault request only by triggering a reset.

r. If ITR field of the SWT_CR register is set to zero, the request is sent on the first time-out. If ITR field is set to one, the request is sent on the second consecutive time-out.

3.8.5 First Timeout event of SWT3 (Fault #100)

3.8.6 First Timeout event of SWT2 (Fault #101)

3.8.7 First Timeout event of SWT1 (Fault #102)

3.8.8 First Timeout event of SWT0 (Fault #103)

If the SWT reaches a timeout<XREF>, the HW forwards this fault to FCCU.

The user can inject this fault by a SW procedure that enables the SWT0 and doesn't service it.

3.9 MEMU Faults

The MEMU is responsible for collecting and reporting error events captured by ECC logic. When an ECC error event occurs, the MEMU receives an error signal and – as consequence - it records the event. The MEMU filters and then forwards notifications in an aggregated manner to the FCCU.

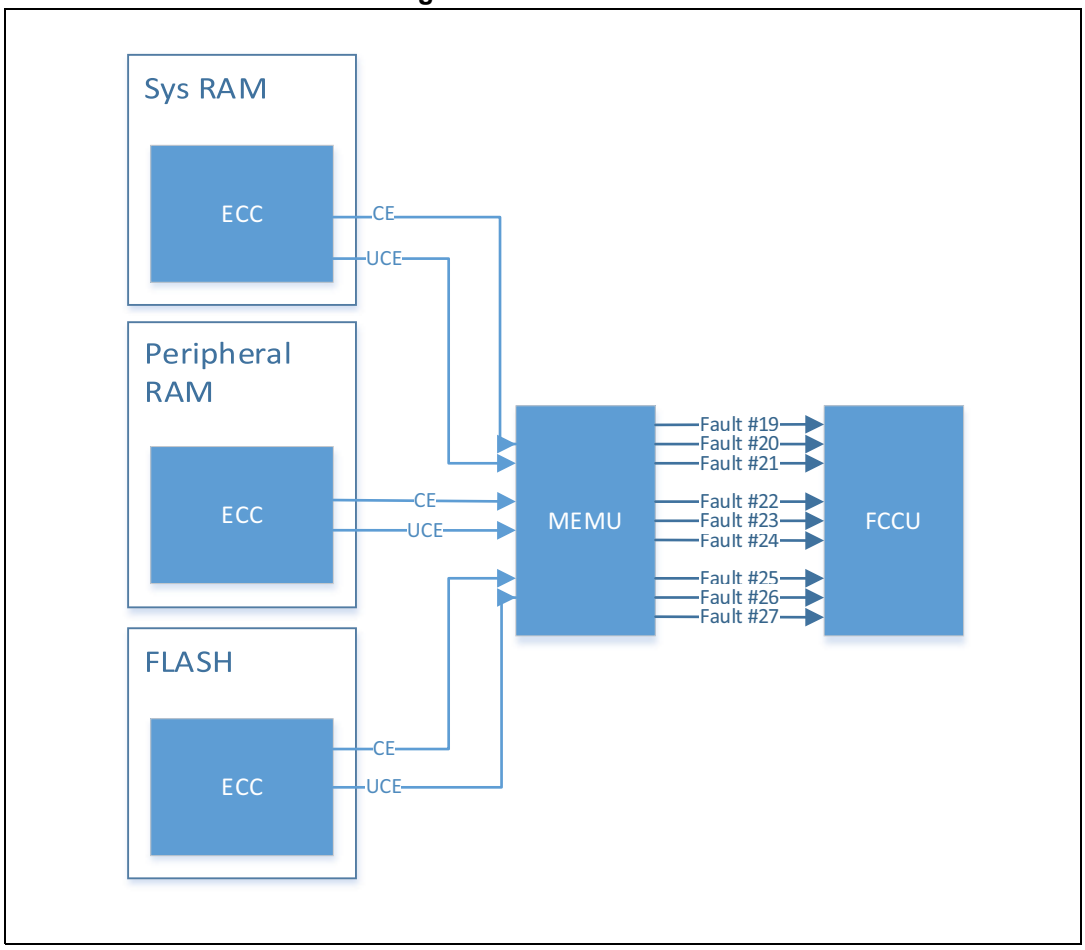
The MEMU does not receive any error signal on ECC events occurring while accessing the EEPROM sections of the FLASH memory.

Note that to clear an FCCU fault related to the MEMU, the user must clear before the corresponding bit in the MEMU ERR_FLAG register[(s)] and then resetting the corresponding bit in the FCCU FCCU_RF_Sn register.

The device provides two mechanisms for accessing any chip memory to read or to modify data including ECC check bits. This capability is useful for test activities, including injecting the FCCU faults related to the MEMU. These mechanisms are the IMA module and the CPU End2End ECC test feature.

s. The MEMU module remains "on" during functional reset.

Figure 11. MEMU Faults



3.9.1 MEMU RAM correctable error (Fault #19)

In the case of correctable errors detected by the ECC logic of a master of the system, the MEMU forwards this fault to the FCCU.

The user can inject this fault by a SW procedure that uses the MEMU DEBUG register.

3.9.2 MEMU RAM uncorrectable error (Fault #20)

In the case of uncorrectable errors detected by the ECC logic of a master of the system, the MEMU forwards this fault to the FCCU.

The user can inject this fault by a SW procedure that uses the MEMU DEBUG register.

3.9.3 MEMU RAM overflow error (Fault #21)

In the case of overflow of the system RAM error reporting table, the MEMU forwards this fault to the FCCU.

The user can inject this fault by a SW procedure that uses the MEMU DEBUG register.

3.9.4 MEMU Peripheral RAM correctable error (Fault #22)

In the case of correctable errors in a peripheral RAM, the MEMU forwards this fault to the FCCU.

The user can inject this fault by a SW procedure that uses the MEMU DEBUG register.

3.9.5 MEMU Peripheral RAM uncorrectable error (Fault #23)

In the case of uncorrectable errors in a peripheral RAM, the MEMU forwards this fault to the FCCU.

The user can inject this fault by a SW procedure that uses the MEMU DEBUG register.

3.9.6 MEMU Peripheral RAM overflow (Fault #24)

In the case of overflow of the peripheral RAM error reporting table, the MEMU forwards this fault to the FCCU.

The user can inject this fault by a SW procedure that uses the MEMU DEBUG register.

3.9.7 MEMU FLASH correctable error (Fault #25)

In the case of correctable errors in FLASH, the MEMU forwards this fault to the FCCU.

The user can inject this fault by a SW procedure that uses the MEMU DEBUG register.

3.9.8 MEMU FLASH uncorrectable error (Fault #26)

In the case of uncorrectable errors in FLASH, the MEMU forwards this fault to the FCCU.

The user can inject this fault by a SW procedure that uses the MEMU DEBUG register.

3.9.9 MEMU FLASH overflow error (Fault #27)

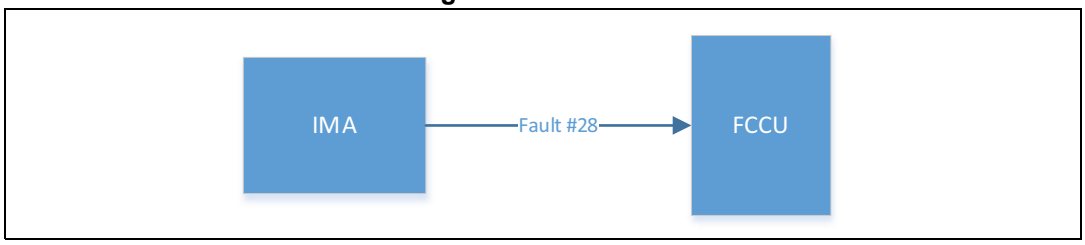
In the case of overflow of the peripheral FLASH error reporting table, the MEMU forwards this fault to the FCCU.

The user can inject this fault by a SW procedure that uses the MEMU DEBUG register.

3.10 IMA Fault

Indirect Memory Access (IMA) refers to the activity of accessing any chip memory for reading and/or modifying data and ECC check bits. This capability is useful for test activities (e.g., verifying the integrity of the ECC logic injecting ECC errors).

Figure 12. IMA Fault



3.10.1 IMA SoC Active (Fault #28)

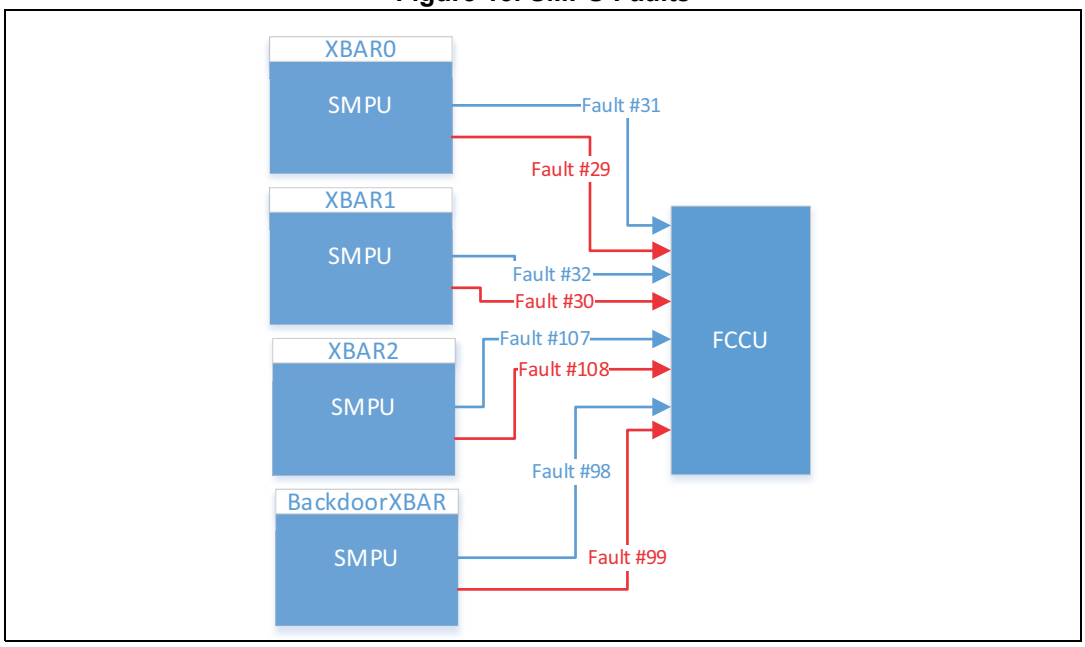
Since unwanted activation of the IMA can interfere with the execution of the safety function, the FCCU monitors unwanted activation of the IMA. Before any intentional access to the memories by the IMA, the user shall disable the relevant FCCU error input.

The user can inject this fault by a SW procedure that uses IMA to access a memory address without disabling the relevant FCCU error input.

3.11 SMPU Faults

The SMPU provides hardware access control for system bus resources. The SMPU concurrently monitors and evaluates system bus transactions using pre-programmed region descriptors that define memory spaces and their access rights. Memory accesses that have sufficient access control rights are allowed to complete, while accesses that are not mapped to any region descriptor or have insufficient rights terminate with an access error response.

Figure 13. SMPU Faults



3.11.1 SMPU XBAR 0 correctly refuses an access (Fault #31)

3.11.2 SMPU XBAR 1 correctly refuses an access (Fault #32)

3.11.3 SMPU XBAR 2 correctly refuses an access (Fault #107)

3.11.4 SMPU Backdoor XBAR correctly refuses an access (Fault #98)

In case of an access to a memory location not mapped to any region descriptor or with insufficient rights, it terminates with an access error response. The SMPU forwards this error to the FCCU.

The user can inject this fault by a SW procedure that accesses a memory address not allowed by SMPU configuration.

3.11.5 SMPU XBAR 0 incorrectly refuses an access (Fault #29)

3.11.6 SMPU XBAR 1 incorrectly refuses an access (Fault #30)

3.11.7 SMPU XBAR 2 incorrectly refuses an access (Fault #108)

3.11.8 SMPU Backdoor XBAR incorrectly refuses an access (Fault #99)

The SMPU include a hardware monitors that detects the following unexpected behavior:

- The SPMU allows an access to a memory location not mapped to any descriptor or with insufficient rights
  and
- The SPMU doesn't allow a memory access with sufficient rights

If this hardware monitor detects a fault, it forwards this event to the FCCU.

The user can't inject this fault.

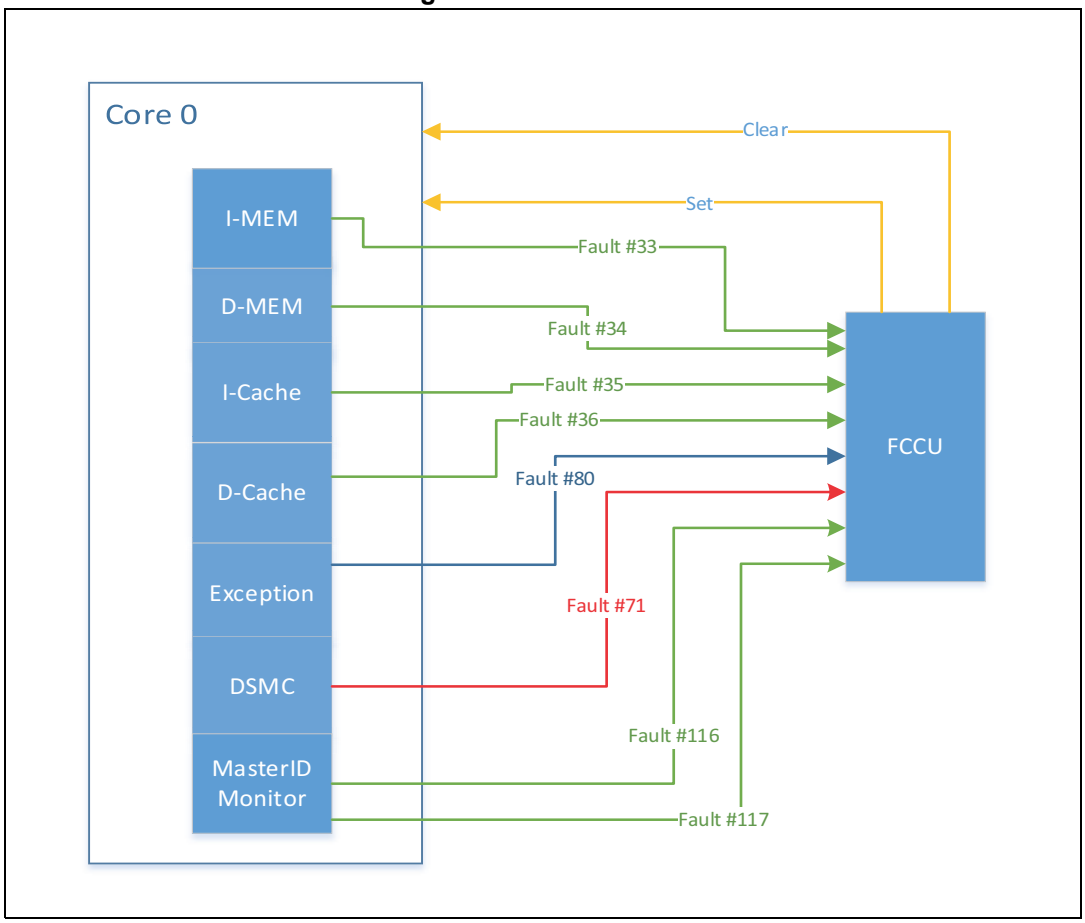
3.12 CORE 0 Faults

Memory feedback checkers inside the core 0 can detect failures affecting the control logic of the memory.

The control signals, which the memory controller sends to the memory array for any operation, are latched and sent back from the memory array to the memory controller. The memory controller compares the received control signals with the transmitted control ones. If they do not match, it forwards an error event to the FCCU.

The Master ID monitor verifies if a master accesses the bus with an erroneous ID.

Figure 14. Core 0 faults



3.12.1 Core 0 I-MEM address feedback checker error (Fault #33)

If the control signals latched back to the memory controller don't match the ones sent to I-MEM bank, the hardware forwards this fault to the FCCU. The user can inject this fault by the FCCU fake fault interface.

3.12.2 Core 0 D-MEM address feedback checker error (Fault #34)

If the control signals latched back to the memory controller don't match the ones sent to D-MEM, the hardware forwards this fault to the FCCU. The user can inject this fault by the FCCU fake fault interface.

3.12.3 Core 0 I-CACHE address feedback checker error (Fault #35)

In case control signals latched back to the memory controller don't match the ones sent to I-CACHE, the hardware forwards this fault to the FCCU. The user can inject this fault by the FCCU fake fault interface.

3.12.4 Core 0 D-CACHE address feedback checker error (Fault #36)

If the control signals latched back to the memory controller don't match the ones sent to D-CACHE, the hardware forwards this fault to the FCCU. The user can inject this fault by the FCCU fake fault interface.

3.12.5 Core 0 DSMC control signal error (Fault #71)

The DSMC generates atomic read-modify-write bus transactions to the attached slave memory controller, the two transactions (read, write) are pipelined without introducing any idle cycle.

The module includes error checking logic that reports to the FCCU a fault affecting the control signals. The user can’t inject this fault.

3.12.6 Core 0 Machine check exception indication (Fault #80)

When the core goes into the machine check condition, the hardware signals this fault to the FCCU. The user can inject this fault by a SW procedure that - for example - accesses a memory affected by an uncorrectable error (e.g., the FLASH address in the Customer Programmable Detection area in the UTEST block containing uncorrectable errors).

3.12.7 Core 0 Master-ID monitor for Data bus error (Fault #116)

When the core tries to access the data bus with an erroneous master ID, the hardware signals this fault to the FCCU. The user can inject this fault by the FCCU fake fault interface.

3.12.8 Core 0 Master-ID monitor for Instruction bus error (Fault #117)

When the core tries to access the instruction bus with an erroneous master ID, the hardware signals this fault to the FCCU. The user can inject this fault by the FCCU fake fault interface.

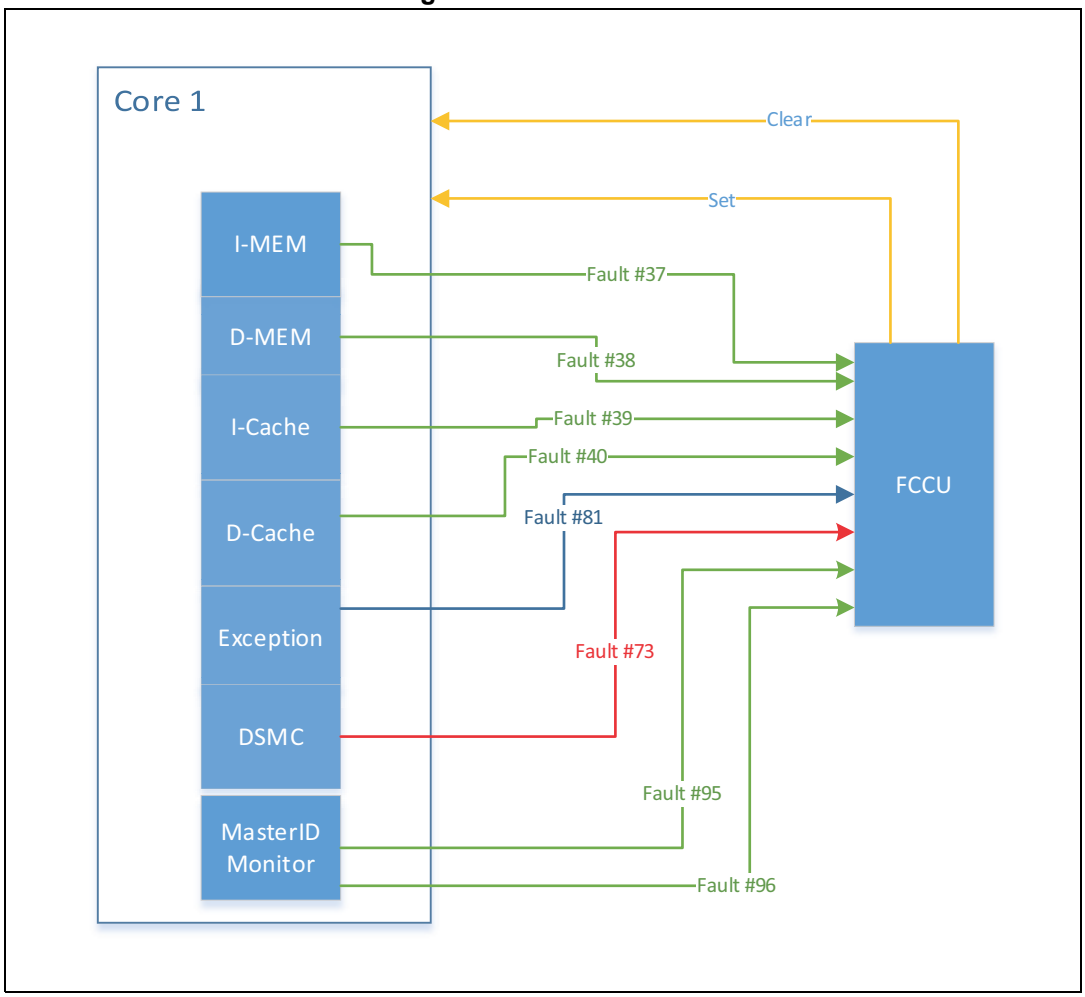
3.13 CORE 1 Faults

Memory feedback checkers inside the core 1 can detect failures affecting the control logic of the memory.

The control signals, which the memory controller sends to the memory array for any operation, are latched and sent back from the memory array to the memory controller. The memory controller compares these received control signals with the transmitted control signals. If they do not match, it forwards an error event to the FCCU.

The Master ID monitor is intended to avoid a master from accessing the bus with a wrong ID.

Figure 15. Core 1 faults



3.13.1 Core 1 I-MEM address feedback checker error (Fault #37)

If the control signals latched back to the memory controller don't match the ones sent to I-MEM, the hardware forwards this fault to the FCCU. The user can inject this fault by the FCCU fake fault interface.

3.13.2 Core 1 D-MEM address feedback checker error (Fault #38)

If the control signals latched back to the memory controller don't match the ones sent to D-MEM, the hardware forwards this fault to the FCCU. The user can inject this fault by the FCCU fake fault interface.

3.13.3 Core 1 I-CACHE address feedback checker error (Fault #39)

If the control signals latched back to the memory controller don't match the ones sent to I-CACHE, the hardware forwards this fault to the FCCU. The user can inject this fault by the FCCU fake fault interface.

3.13.4 Core 1 D-CACHE address feedback checker error (Fault #40)

If the control signals latched back to the memory controller don't match the ones sent to D-CACHE, the hardware forwards this fault to the FCCU. The user can inject this fault by the FCCU fake fault interface.

3.13.5 Core 1 DSMC control signal error (Fault #73)

The DSMC generates atomic read-modify-write bus transactions to the attached slave memory controller, the two transactions (read, write) are pipelined without introducing any idle cycle. During the entire read-modify-write process, the slave can't satisfy any other read or write request from a master.

The module includes error checking logic that reports to the FCCU a fault affecting the control signals. The user can’t inject this fault.

3.13.6 Core 1 Machine check exception indication (Fault #81)

When the core goes into the machine check condition, the hardware signals this fault to the FCCU. The user can inject this fault by a SW procedure that - for example - accesses a memory affected by an uncorrectable error (e.g., the FLASH address in the Customer Programmable Detection area in the UTEST block containing uncorrectable errors).

3.13.7 Core 1 Master-ID monitor for Data bus error (Fault #95)

When the core tries to access the data bus with an erroneous master ID, the hardware signals this fault to the FCCU. The user can inject this fault by the FCCU fake fault interface.

3.13.8 Core 1 Master-ID monitor for Instruction bus error (Fault #96)

When the core tries to access the instruction bus with an erroneous master ID, the hardware signals this fault to the FCCU. The user can inject this fault by the FCCU fake fault interface.

3.14 CORE 2 Faults

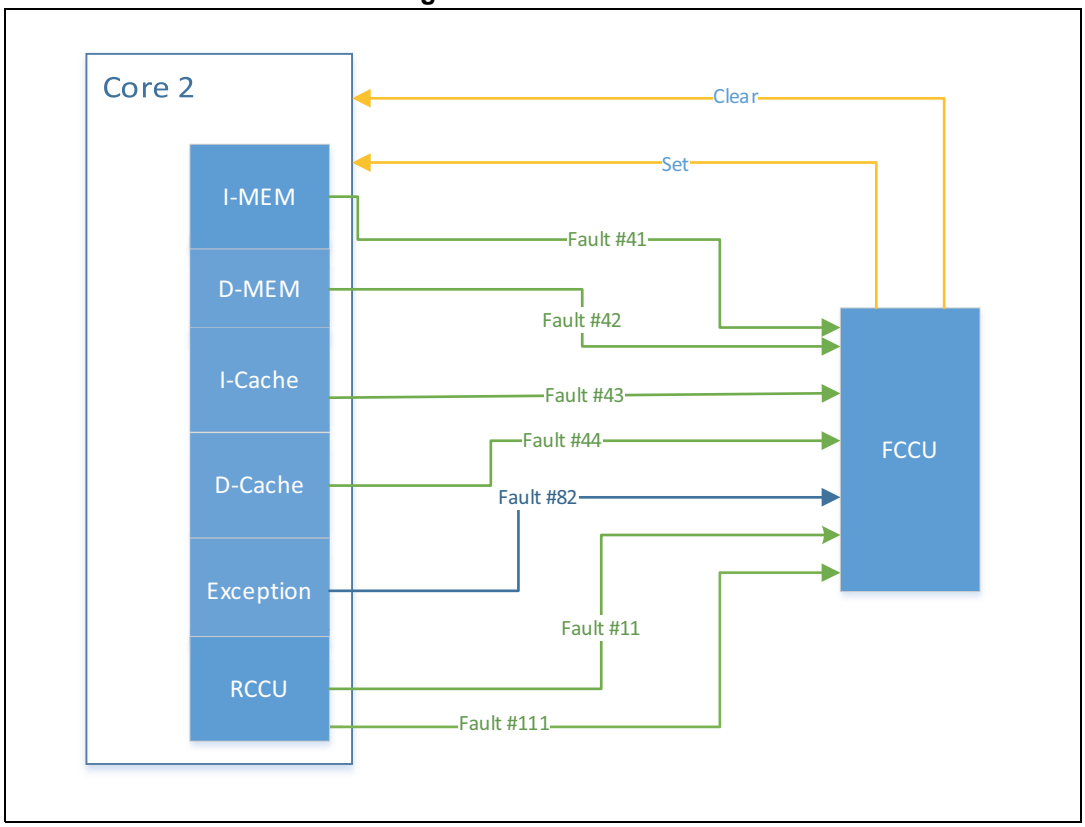
Memory feedback checkers inside the core 2 are able detect failures affecting the control logic of the memory.

The control signals, which the memory controller sends to the memory array for any operation, are latched and sent back from the memory array to the memory controller. The memory controller compares these received control signals with the transmitted control signals. If they do not match, it forwards an error event to the FCCU.

The core 2 and its strictly connected modules are replicated. The RCCU compares the operation of the safety core and its replica. If it detects a fault, it triggers an event to the FCCU.

Besides, the FCCU monitor the activation of the replicated core (i.e., LSM).

Figure 16. Core 2 faults



3.14.1 Lockstep out of sync in the Core 2 or associated DSMC (Fault #11)

In the case of a mismatch between the operation of the safety core and its replica, the RCCU signals the fault to the FCCU. The user can inject this fault by the FCCU fake fault interface.

3.14.2 Core 2 I-MEM address feedback checker error (Fault #41)

In case control signals latched back to the memory controller don't match the ones sent to I-MEM, the hardware forwards this fault to the FCCU. The user can inject this fault by the FCCU fake fault interface.

3.14.3 Core 2 D-MEM address feedback checker error (Fault #42)

In case control signals latched back to the memory controller don't match the ones sent to D-MEM, the hardware forwards this fault to the FCCU. The user can inject this fault by the FCCU fake fault interface.

3.14.4 Core 2 I-CACHE address feedback checker error (Fault #43)

In case control signals latched back to the memory controller don't match the ones sent to I-CACHE, the hardware forwards this fault to the FCCU. The user can inject this fault by the FCCU fake fault interface.

3.14.5 Core 2 D-CACHE address feedback checker error (Fault #44)

In case control signals latched back to the memory controller don't match the ones sent to D-CACHE the hardware forwards this fault to the FCCU. The user can inject this fault by the FCCU fake fault interface.

3.14.6 Core 2 Machine check exception indication (Fault #82)

When the core goes into the machine check condition, the hardware signals this fault to the FCCU. The user can inject this fault by a SW procedure that - for example - accesses a memory affected by an uncorrectable error (e.g., the FLASH address in the Customer Programmable Detection area in the UTEST block containing uncorrectable errors)

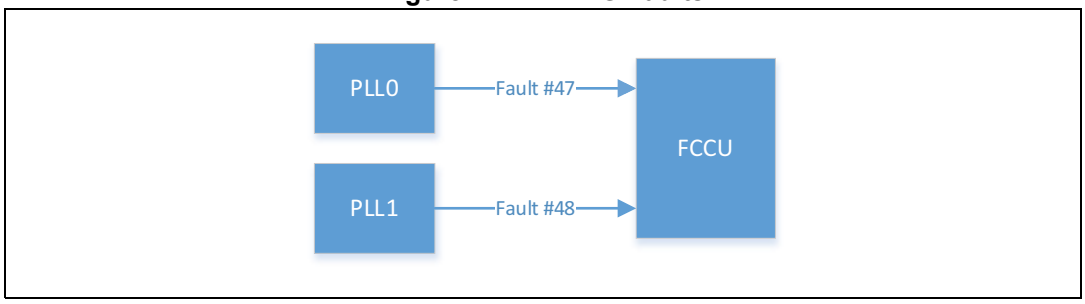
3.14.7 Core 2 Indication of disablement of lockstep mode (Fault #111)

In case of disablement of the LSM on the safety core, the event is forwarded to the FCCU. The user can inject this fault by the FCCU fake fault interface.

3.15 PLLDIG Faults

The SPC58EHx/SPC58NHx embeds a dual PLL system which delivers separate system and peripheral clocks.

Figure 17. PLL DIG Faults



3.15.1 PLL 0/1 Loss of lock (Fault #47 and Fault #48)

A built-in mechanism can detect a loss of lock for the PLL0 and the PLL1. The PLLDIG forwards this fault to the FCCU. The user can inject this fault by a SW procedure that changes on-the-fly the PLL configuration that generates a loss of lock.

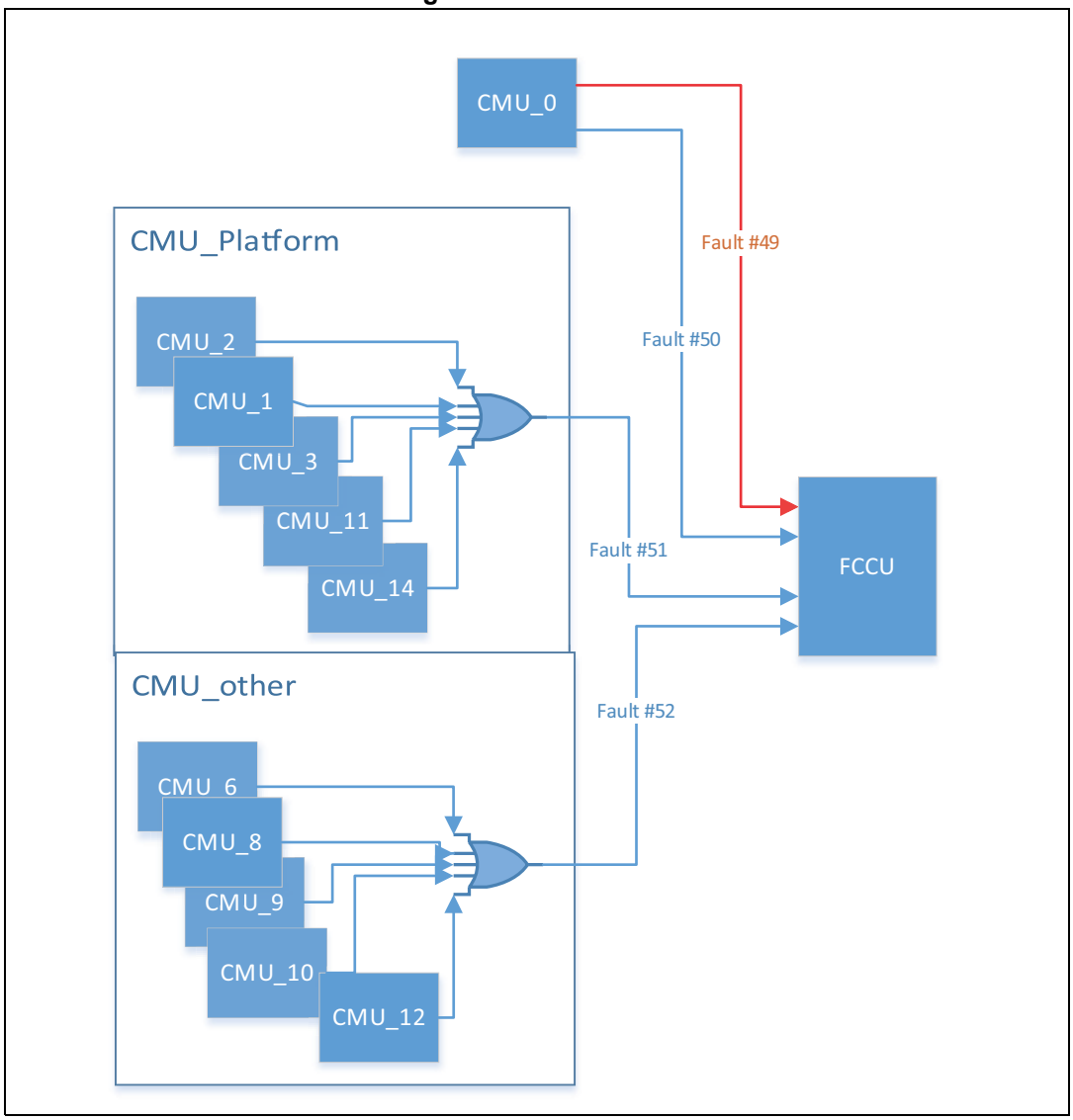
3.16 CMU Faults

Different CMU modules supervise the integrity of the clock sources of the device. If the monitored clock frequency is outside the expected frequency range, the CMU detects and forwards[t] this event to the FCCU. Refer to device SPC58EHx/SPC58NHx Microcontroller Reference Manual for further details on CMU configuration an which clock is monitored[u].

t. The user must enable the CMU that is disabled by default.

u. Refer to the section “Clock input sources” of the reference manual.

Figure 18. CMU Faults



3.16.1 XOSC less than IRC, loss of XOSC clock (OLR signal from CMU0) (Fault #49)

The CMU_0 monitors the XOSC frequency. If the XOSC frequency is less than a configurable value or it is not stable (e.g., the XOSC is not connected), the CMU_0 can detect this event and forwards it to the FCCU.

The user can't inject this fault[v].

v. The CMU0 checks if the Frequency of the XOSC is less than the frequency of the IRC divided by a configurable factor within a range from 1 to 8. If the frequency of the XOSC is higher than the frequency of the IRC it is not possible inject the fault by a SW procedure.

3.16.2 System clock frequency out of range (Fault #50)

The CMU_0 monitors the system clock frequency (PHI output of PLL_0) using the IRCOSC and XOSC frequencies as monitor references. If the PLL_0 output frequency is above or below the monitoring thresholds, the CMU_0 can detect this event and forward it to the FCCU. The user can inject this fault by a SW procedure that sets misconfigured values for the monitoring thresholds.

3.16.3 Platform clock frequency out of range (Fault #51)

The CMU_1 monitors the clock frequency used by COREs, HSM and the XBAR. The CMU_2 monitors the clock frequency used by DMA, SIPI, SWT, STM and INTC. The CMU_3 monitors the clock frequency used by all peripheral bridges and peripherals not connected to any other system clock divider. The CMU_11 monitors the clock frequency used by SEM4. The CMU_14 monitors the clock frequency used by M_CAN.

If one of these frequencies is above or below the monitoring thresholds, the relevant CMU can detect this event and forwards it to the FCCU[w]<XREF>,. The user can inject this fault by a SW procedure that sets misconfigured values for the monitoring thresholds.

3.16.4 Other clocks frequency out of range (Fault #52)

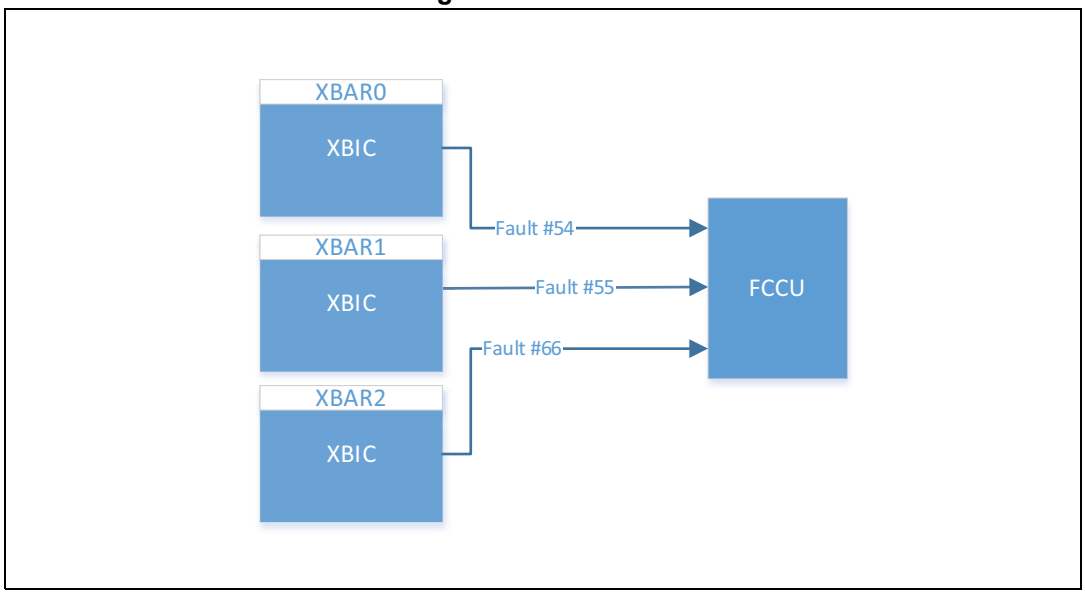
The CMU_6 monitors the clock frequency used by SARADC and the CMU_12 monitors the clock frequency used by EMIOS. The CMU_8, CMU_9 and CMU_10 monitor the clock frequency used by PSI5. If one of these frequencies is out of the monitoring thresholds, the relevant CMU can detect this event and forwards it to the FCCU<XREF>,<XREF>. The user can inject this fault by a SW procedure that sets misconfigured values for the monitoring thresholds.

3.17 XBIC Fault

The XBIC monitors the integrity of each crossbar transfer.

w. The FCCU receives the OR'ed signal from these CMU modules.

Figure 19. XBIC fault



3.17.1 XBAR 0/1/2 XBIC fault (Fault #54, Fault #55 and Fault #66)

In case of a corrupted transaction through the XBAR, the XBIC detects this event and forwards it to the FCCU.

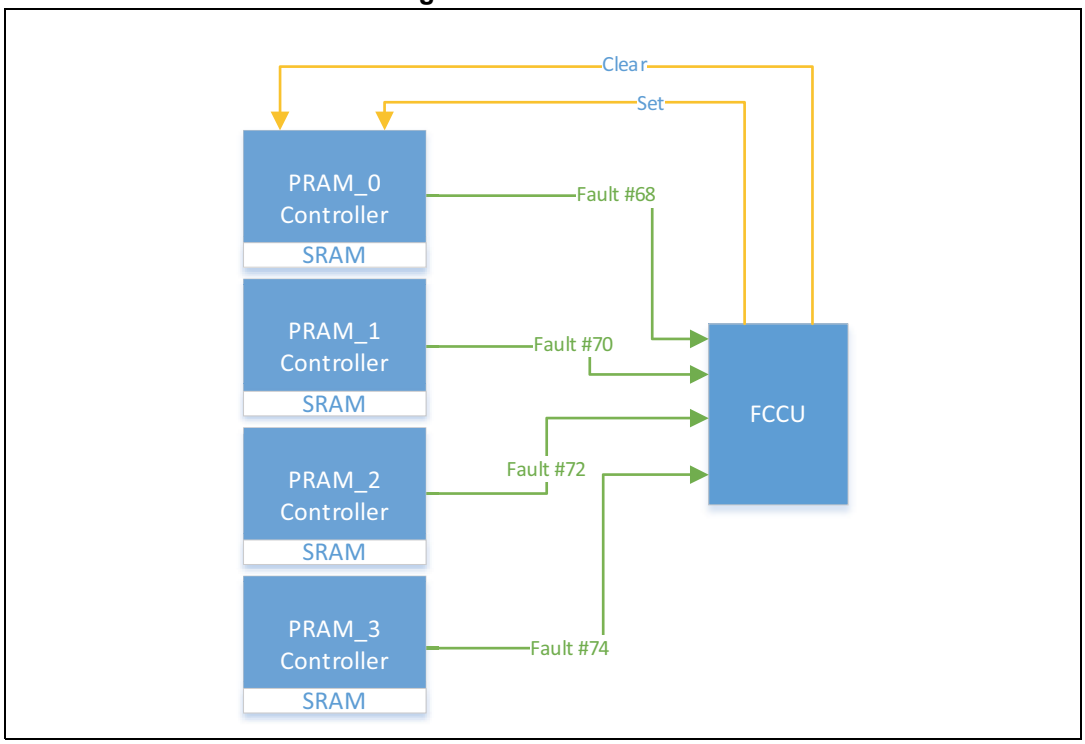
The user can inject this fault by a SW procedure that uses the XBIC error injection feature through the XBIC_EIR register.

3.18 PRAM Faults

The PRAM controller is the interface between the system bus and the system RAM. It converts the protocols between the system bus and the RAM array interface.

The device embeds four controllers.

Figure 20. PRAM Faults



3.18.1 PRAM 0 System RAM Address Feedback error or RAM Late-Write Buffer mismatch error or EDC after ECC for RAM (RMW ECC) error (Fault #68).

3.18.2 PRAM 1 System RAM Address Feedback error or RAM Late-Write Buffer mismatch error or EDC after ECC for RAM (RMW ECC) error (Fault #70).

3.18.3 PRAM 2 System RAM Address Feedback error or RAM Late-Write Buffer mismatch error or EDC after ECC for RAM (RMW ECC) error (Fault #72).

3.18.4 PRAM 3 System RAM Address Feedback error or RAM Late-Write Buffer mismatch error or EDC after ECC for RAM (RMW ECC) error (Fault #74).

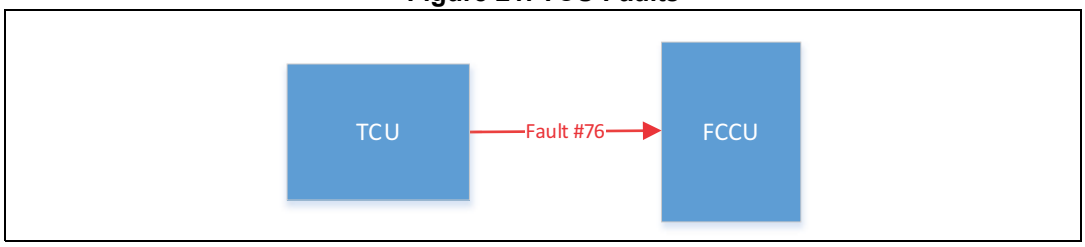
The PRAM controller can detect addressing error through the address feedback mechanism. The EDC after ECC detects a random failure affecting the ECC logic of the PRAM controller during RMW operations. The resulting signal from the OR logic operation of these faults is forwarded to the FCCU.

The user can inject this faults by the FCCU fake fault interface.

3.19 TCU Fault.

The TCU is a module that activates the testing functionalities of the device. An unwanted activation of the test mode can impact the execution of the safety function. For this reason, the hardware monitors the unwanted activation of the test mode. If it occurs, the FCCU receives the indication of the fault.

Figure 21. TCU Faults



3.19.1 Test circuitry activation (Fault #76)

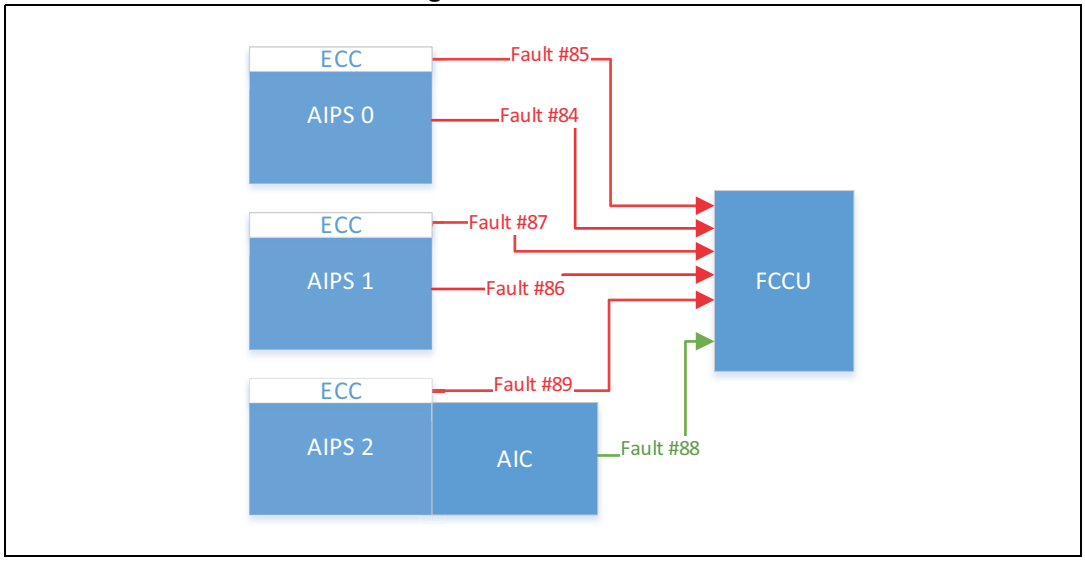
In case of unwanted activation of the test circuitry, the event is detected and forwarded to the FCCU.

The user can't inject this fault.

3.20 AIPS Faults

The peripheral bridge connects the crossbar switch interface to the register interface of the peripherals embedded within the device. The device integrates three peripheral bridges: AIPS_0, AIPS_1 and AIPS_2. Each one of these modules embeds the ECC logic to implement the End-to-End protection on data path (e2eECC). The e2eECC schema provides high detection capabilities against failures affecting the data content of the transaction. The AIPS_2 is also protected from random failure by a lockstep replica that monitors the control signals (i.e., AIC).

Figure 22. AIPS Faults



3.20.1 Frequency error between XBAR 0 and the AIPS 0 (Fault #84)

3.20.2 Frequency error between XBAR 1 and the AIPS 1 (Fault #86)

3.20.3 Frequency error between XBAR 1 and the AIPS 2 or AIC error (Fault #88)

Whether a hardware fault results in a wrong frequency conversion between the XBAR and the AIPS, the hardware triggers an event to the FCCU.

In the case of a corrupted transaction through the AIPS_2, the AIC detects this event and forwards it to the FCCU. The user can't inject frequency error fault. The user – however – can inject the AIC fault by the fake fault interface of the FCCU.

3.20.4 EDC after ECC error on a transaction through the AIPS 0 (Fault #85)

3.20.5 EDC after ECC error on a transaction through the AIPS 1 (Fault #87)

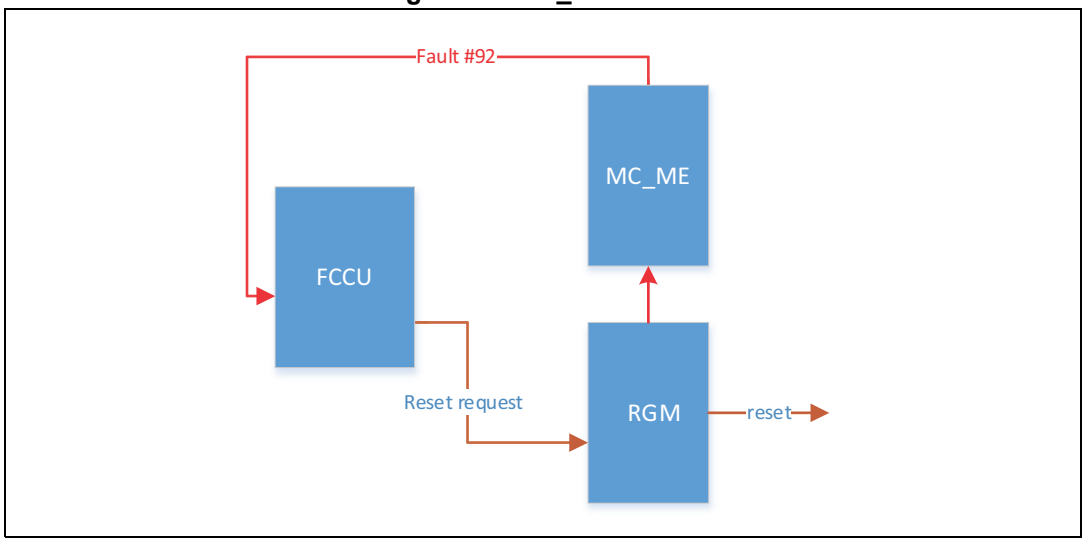
3.20.6 EDC after ECC error on a transaction through the AIPS 2 (Fault #89)

A random failure affecting the ECC correction logic can cause a corrupted ECC correction. The EDC after ECC can detect this event and forward it to the FCCU. The user can't inject this fault.

3.21 MC_RGM Fault

The MC_RGM centralizes the different reset sources and manages the reset sequence of the chip.

Figure 23. MC_RGM Fault



3.21.1 Safe Mode Entry Indication (Fault #92)

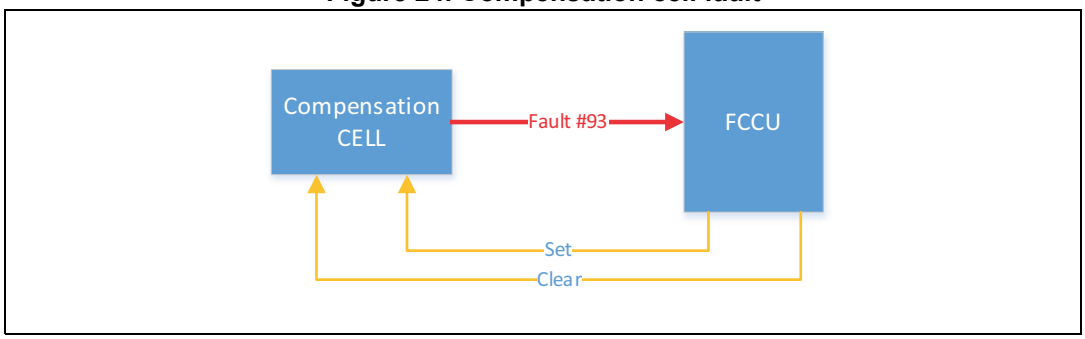
The MC_RGM can request a transition to SAFE mode to the MC_ME. In the case of an unwanted safe mode request due to a random event, the hardware detected and forwards this event to the FCCU.

The user can't inject this fault.

3.22 Compensation cells Fault

Compensation cells generate 8-bit compensation code for IO. It reduces the spread of some circuit parameters[x] in the IO buffers over temperature, pressure and voltage.

Figure 24. Compensation cell fault



3.22.1 Pad Compensation Disabled (Fault #93)

When the compensation cell is in normal mode, it generates the compensation codes. If it exits the normal mode, the hardware detects this event and forwards it to the FCCU. The user can't inject this fault.

3.23 Interrupt Controller (INTC) Faults

The INTC provides a mechanism to schedule and service the interrupt requests external to the cores.

The INTC is duplicated and the operation are executed in lockstep mode. The replicated operations are compared and any operational deviation between the supervised signals notifies the FCCU of the discrepancy.

3.23.1 INTC Lockstep out of sync due to a fault (Fault #12)

In case a deviation in the operation of the INTC and its replica, the RCCU signals the fault to the FCCU.

The user can inject this fault by the FCCU fake fault interface.

x. Some of these parameters are the slew rate of the output signal and the output impedance.

3.23.2 Indication of disablement of INTC lockstep mode (Fault #112)

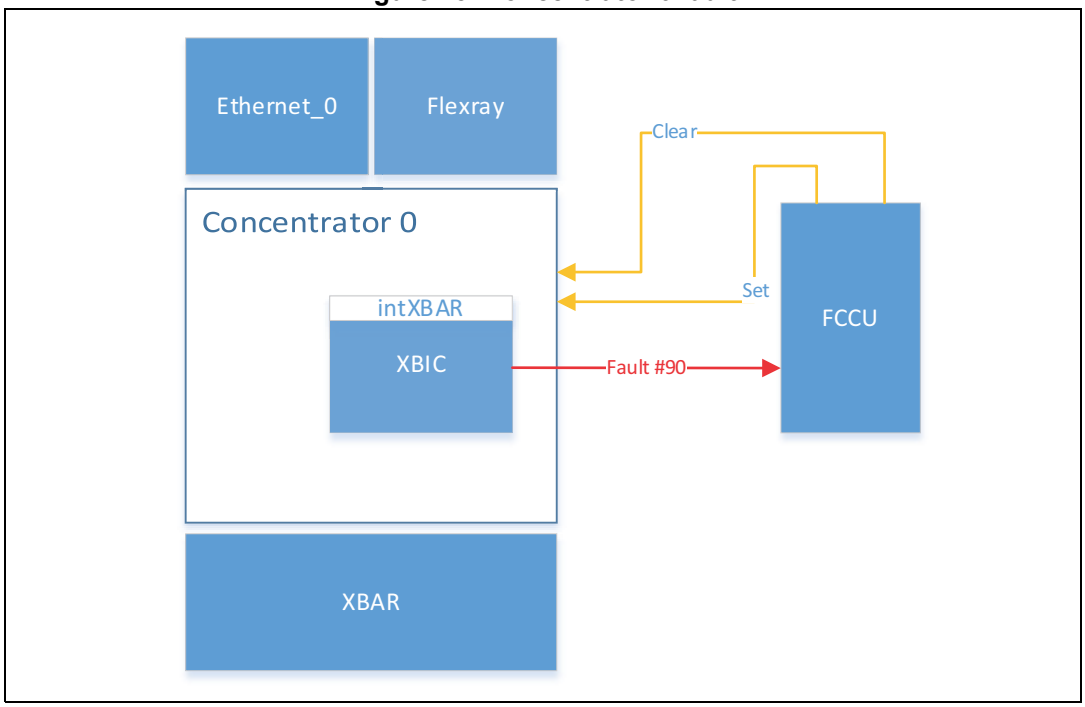
In case of disablement of the LSM in the INTC, the event is forwarded to the FCCU.

The user can inject this fault by the FCCU fake fault interface.

3.24 Concentrator Faults

Two different concentrators exist in the device. They forward and handle data and control signals between the crossbar and some master devices. The concentrator 0 is linked to the Ethernet_0 and the Flexray modules. The concentrator HSM is linked to the HSM core.

Figure 25. Concentrator 0 fault



3.24.1 Concentrator 0 fault (Fault #90)

Concentrator 0 embeds an internal XBAR along with an XBIC. The XBIC checks the integrity of signals that are routed through this XBAR. In case of error, it forwards a fault to the FCCU.

The user can't inject this fault.

3.24.2 Concentrator HSM EDC after ECC fault (Fault #110)

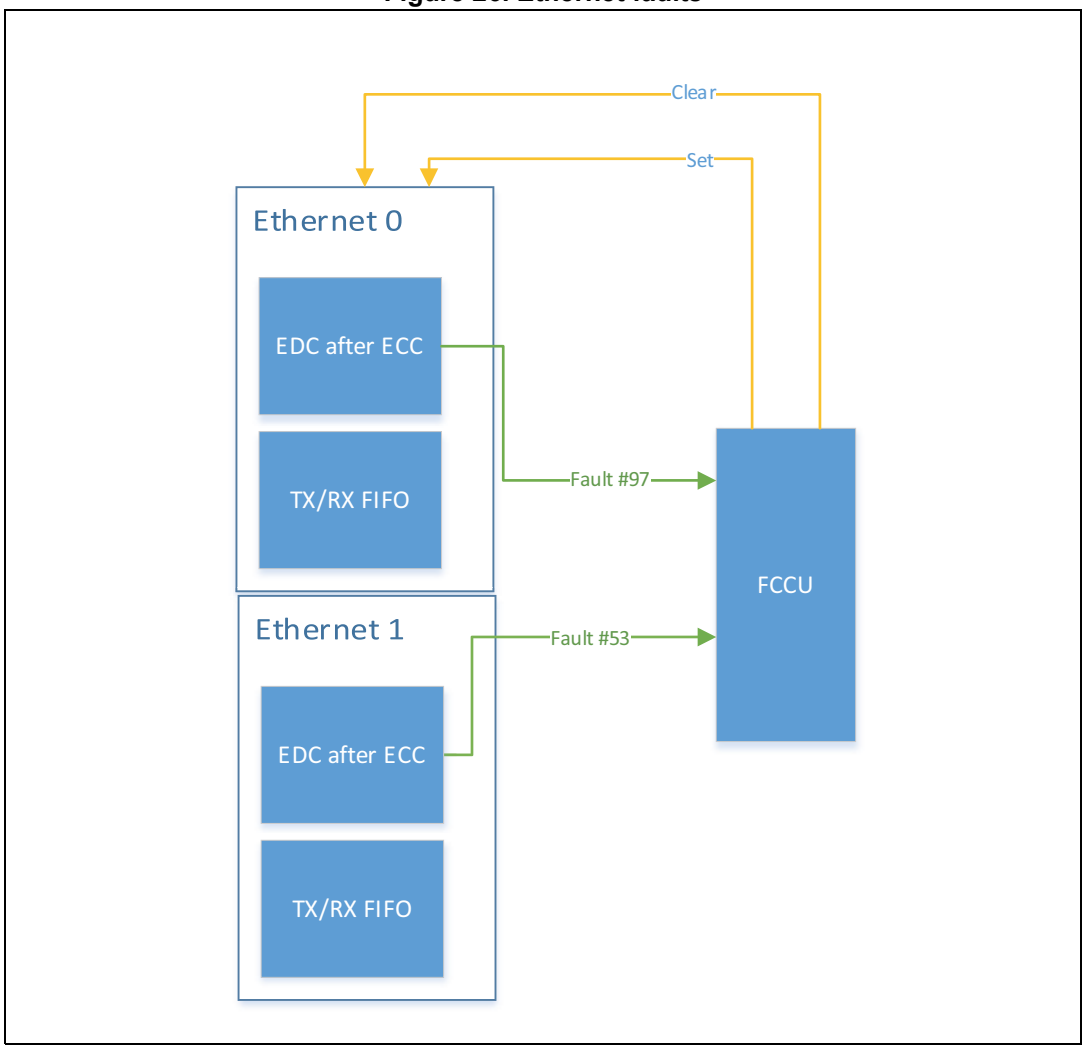
Concentrator HSM embeds an ECC logic. In case of error in the ECC logic, the EDC after ECC forwards a fault to the FCCU.

The user can inject this fault by the FCCU fake fault interface.

3.25 Ethernet Faults

The Ethernet module enables a host to transmit and receive data over the Ethernet in compliance with the IEEE 802.3-2015. The Transmit FIFO (Tx FIFO) and the Receive FIFO (Rx FIFO) stores the Ethernet frames in a RAM protected by single error correction and double error detection ECC scheme.

Figure 26. Ethernet faults



3.25.1 Ethernet 0/1 EDC after ECC error (Fault #97 and Fault #53)

The EDC after ECC signals to the FCCU if a hardware fault in the ECC logic results in a corrupted ECC correction(y).

The user can inject this fault by the FCCU fake fault interface.

y. The data read out of the RAM is correct, but it's modified by the ECC logic due to a random fault.

3.26 Flexray fault

The device embeds a FlexRay communication controller that implements the FlexRay Communications System Protocol Specification, Version 2.1 Rev A. This controller includes two RAM modules:

- Protocol engine data RAM
- Controller host interface (CHI) lookup table RAM

Both of them are protected by SEC/DEC ECC scheme to implement the End-to-End protection on data path (e2eECC).

3.26.1 Flexray EDC after ECC error or Monitor error (Fault #60)

The EDC after ECC signals to the FCCU whether a hardware fault in the ECC logic results in a corrupted ECC correction. The Monitor Error reports an error on a transaction initiated by Flexray master to any slave. The user can inject this fault by the FCCU fake fault interface.

3.27 Octal SPI fault

The OCTOSPI is a specialized communication interface targeting single, dual, quad or octal SPI memories. This module embeds ECC logic to implement the End-to-End protection on data path (e2eECC).

Octal SPI EDC after ECC error (Fault #105)

The EDC after ECC signals to the FCCU whether a hardware fault in the ECC logic results in a corrupted ECC correction.

The user can inject this fault by the FCCU fake fault interface.

3.28 eMMC Fault

The eMMC module interfaces to the system bus using one AHB master and one slave interface. The slave bus accesses the registers inside the Host controller. The master bus is used by the DMA controller (when using DMA or ADMA2 modes). Both interfaces embed the ECC logic to implement the End-to-End protection on the data path (e2eECC).

3.28.1 eMMC Master or Slave EDC after ECC error (Fault #106)

The EDC after ECC signals to the FCCU if a hardware fault in the ECC logic results in a corrupted ECC correction.

The user can inject this fault by the FCCU fake fault interface.

3.29 USB fault

Only the Cut 1 of the device embeds an USB module. The USB module interfaces to the system bus using one AHB master and one slave interface. Both interfaces embed the ECC logic to implement the End-to-End protection on the data path (e2eECC).

3.29.1 USB Master/Slave EDC after ECC error or Master ECC error (Fault #104)

An ECC error is reported to on the master interface is reported to the FCCU. The EDC after ECC signals to the FCCU if a hardware fault in the ECC logic, for master or slave interface, results in a corrupted ECC correction.

The user can inject this fault by the FCCU fake fault interface.

3.30 Zipwire faults

Zipwire is a group of modules that allow one MCU to have a fast, low pin count, serial communication link directly into the memory mapped peripherals or memories of another MCU (or both) or smart ASIC. Only the Cut 1 of the device embeds two Zipwire modules.

3.30.1 Zipwire 0 EDC after ECC error (Fault #113)

The EDC after ECC signals to the FCCU if a hardware fault in the ECC logic results in a corrupted ECC correction.

The user can inject this fault by the FCCU fake fault interface.

3.30.2 Zipwire 1 EDC after ECC error (Fault #114)

The EDC after ECC signals to the FCCU if a hardware fault in the ECC logic results in a corrupted ECC correction.

The user can inject this fault by the FCCU fake fault interface.

4 Example code

An example code that includes the FCCU settings and how to inject FCCU faults according to the above list is available upon request. Hereafter a summary of the tasks done in the example code:

1. Initialize the MCU (clocks and modules);
   a) Reset the RGM and clear its registers
   b) Initialize the FCCU and configure all testable faults as described below:

      They are enabled
      They are SW recoverable
      No reset action
      Interrupt is enabled
      Output pins are enabled
      FCCU state machine goes to Alarm state in case of fault

   c) For each testable faults the software:

      Verify the FCCU status before the injection of the fault:

      If in Normal state, proceed

      If in Alarm or Fault state, stop

      Inject the fault (using the monitor's registers, or using the fake fault injection, or a SW procedure)
      Verify the FCCU status after injection:

      If in Normal state, proceed

      If in Alarm or Fault state, stop

      Clear the monitor and FCCU Alarm state
      Verify the FCCU status after recovering from Alarm:

      If in Normal state, proceed

      If in Alarm or Fault state, stop

      Check FCCU reaction (IRQ, Reset Request, EOUT)

5 Summary

The safety analysis requires that the user verifies the integrity of the error reaction path periodically with a period of a trip time which is in the order of 12 hours. The user must verify the entire error reaction path including the connection between the monitors and the FCCU[z] and the FCCU reactions. The methodology for these tests depends on the specific FCCU input. The idea, however, is to inject a fault and to verify if whether the FCCU correctly receives and react[aa] to it.

This document - concerning SPC58EHx/SPC58NHx devices - describes the FCCU faults inputs and how to verify their reaction path.

z. Not all FCCU inputs are testable. User must test only the enabled ones.

aa. The user doesn't need to check the FCCU reaction for all testable faults. It's enough to test the FCCU reaction only for a single fault.

6 Revision history

Table 2. Document revision history

Date	Revision	Changes
08-Feb-2019	1	Initial release.